**MACHINE LEARNING AND OPTIMIZATION-BASED MODELING FOR ASSET MANAGEMENT**

by
Carlos Yohan Rafavy
Bachelor of Engineering in Industrial Engineering
Atma Jaya Catholic University of Indonesia, 2013
and
Justin Patrick Casey
Bachelor of Science in Accounting and Finance
University of Kentucky, 2012

SUBMITTED TO THE PROGRAM IN SUPPLY CHAIN MANAGEMENT
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
**MASTER OF APPLIED SCIENCE IN SUPPLY CHAIN MANAGEMENT**
AT THE
**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**
May 2020



Signature of Author_____
Carlos Yohan Rafavy
Department of Supply Chain Management
May 8, 2020

Signature of Author_____
Justin Patrick Casey
Department of Supply Chain Management
May 8, 2020

Certified by _____
Dr. Andrés Felipe Muñoz-Villamizar
Postdoctoral Associate, Center for Transportation and Logistics
Capstone Advisor

Accepted by _____
Dr. Yossi Sheffi
Director, Center for Transportation and Logistics
Elisha Gray II Professor of Engineering Systems
Professor, Civil and Environmental Engineering

**MACHINE LEARNING AND OPTIMIZATION-BASED MODELING FOR ASSET MANAGEMENT**

by

Carlos Yohan Rafavy

and

Justin Patrick Casey

Submitted to the program in Supply Chain Management

on May 8, 2020 in Partial Fulfilment of the Requirements for the Degree of

Master of Applied Science in Supply Chain Management

at the

Massachusetts Institute of Technology

**ABSTRACT**

This capstone project is sponsored by a water technology company and particularly covers its industrial pump rental business across the United States. With millions of dollars of annual spending for pump mobilization, the company looks for ways to improve the overall asset utilization rate. At its current practice, the company has not regularly used any statistical method or algorithm for demand prediction. Moreover, decisions for asset movement between branches are largely arranged between individual branch managers on an as-needed basis. We propose an improvement for the company's asset management practice by modeling an integrated decision tool which involves evaluation of several machine learning algorithms for demand prediction and mathematical optimization for a centrally-planned asset allocation. We find that a feed-forward neural network (FNN) model with single hidden layer is the best performing predictor for the company's intermittent product demand and the optimization model is proven to prescribe the most efficient asset allocation given the demand prediction from FNN model.

Capstone Advisor : Dr. Andrés Felipe Muñoz-Villamizar

Title : Postdoctoral Associate, MIT Center for Transportation and Logistics

## ACKNOWLEDGEMENTS

We would like to thank our advisor Dr. Andrés Felipe Muñoz-Villamizar for his patience, constructive feedbacks, insights, and guidance which have driven us towards the accomplishment of this project. Many of the staffs at MIT Center for Transportation and Logistics have also provided countless support and encouragement during the many dynamic and demanding periods at MIT when we sometimes wished there were more than twenty four hours in a day. Definitely not the least is our gratitude to the special SCM class of 2020, the bond that has been forged out of our time together both in and out of classes, though had to be cut prematurely due to the extraordinary pandemic that we are going through, will certainly extend beyond our farewell where ever each of us ends up in around the globe. For having met, known, and worked with all of you, we are honored.

To the stakeholders in the sponsoring company of this capstone project, we would like to present our thanks and appreciation for the collaboration throughout the duration of the project. We hope that the decision tool we developed for you becomes a significant contribution in the company's ongoing journey towards a better data-driven asset management practice.
*Carlos Yohan Rafavy & Justin Patrick Casey*

I dedicate this report and the completion of my degree to my mother, Rospita Parapat, and my father, Charlon Situmeang, who have long instilled in myself a lasting passion for learning, independence, and work ethics. Special acknowledgements to Nicolas Denis, Athanasios Katrantzis, and Dimitris Takvorian of Philip Morris International whose valuable support, mentorship, and endorsement have made my lifelong dream of studying at MIT a fulfilled reality. To my capstone partner, Justin, thank you for an enjoyable collaboration and great work.
*Carlos Yohan Rafavy*

Thank You – Friends, Family, and Classmates. And to my coauthor of this capstone report, it has been a pleasure working with you. Happy reading.
*Justin Patrick Casey*

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# 1. INTRODUCTION

## 1.1 Background and Motivation

The company sponsoring our capstone project specializes in water technology and operates globally. It designs and manufactures equipment as well as provides customized service solutions for applications in various stages of water usage cycle. During the most recent fiscal year, almost half of the company's annual revenue came from its water infrastructure business segment alone. This segment covers the transport and treatment of water from a source for use in public utility and industry facilities. It also covers provision of transport solutions and equipment for removal of water from industrial facilities, construction sites, public infrastructure, and utilities in various repair, maintenance, natural disaster, and other emergency circumstances. Within this segment, the company is sponsoring our research project on the industrial pump rental business line, particularly its operations in the United States.

The company's equipment rental business comprises a fleet of around 6,000 units of pumps and 1,200 professionals across 5 regions and 47 branch locations in the United States. It primarily serves dewatering needs, which involves removal of water from solid material or soil, of the construction company, oil, gas, and mining industries as well as public utilities in emergency situations. As such, the majority of incoming demands require fast response, and potential sales will be lost if relevant equipment is not readily available. In general, the company sets an aggressive target to fulfil demand for a turnkey solution in such a situation.

Each year the company spends around millions of dollars in logistic costs to mobilize its rental fleet among the branches in its network of service areas. The iterative distribution effort is aimed at primarily providing satisfactory service level to its customers as well as identifying rooms for improvement in the dewatering assets' utilization rate. In a cursory analysis of the company's lost sales record in the most recent year we found that, out of instances with recorded reasons, 9% of lost sales were related to lack of equipment availability at a branch, which represents 6% lost revenue. If we include lost sales due to competitive reasons, the figures balloon to 24% and 36% out of the total number of lost sales instances and the total amount of lost revenue respectively.

## 1.2 Problem Statement

The company is looking for ways to increase utilization of its rental pump fleet in terms of original equipment cost (OEC) while at the same time keeping the cost of fleet mobilization as efficient as possible. So far, the company has utilized neither a statistical method to forecast demand nor a centrally optimized fleet allocation across its branches.

We identified an improvement opportunity which would propel the company towards a more data-driven demand planning and asset management practice. To that end, we propose the development of demand forecasting through machine learning models based on historical rental sales data and exogenous factors. In addition, a mathematical optimization model to determine the most efficient asset movement in each planning period (month) is also developed. Those two models are integrated in the final delivery to the company in the form of a decision tool written in Python programming language. The tool takes in asset stock data from the most recent month and predicted demand for the subsequent month to produce a set of asset allocation decision. The methodology to develop this tool is divided into four stages: Data Preparation, Predictive Modeling, Optimization Modeling, and Model Integration. Our results show a superior prediction performance by a feed-forward neural network model and an efficient allocation decision prescribed by the optimization model. Therefore, it is expected that the use of this new tool will close the gap between the company's current and desired future level of operational performance and consequently increase its competitiveness

## 2. LITERATURE REVIEW

### 2.1 Industrial Equipment Rental Industry and Water Infrastructure

The scope of this capstone project is the pump rental business of the sponsoring company in the United States. A recent industry analysis report (Roth, 2019) lists pump rental as a subset of the more general industrial equipment rental industry with a predicted annual growth of 2.3% in the 5-year period from 2019. Industrial pumps are among the types of equipment rented by the biggest market in the industry, which comprises manufacturers and other heavy industrial companies, followed by the construction sector and other sectors, including government and institutional clients. The report identifies several general economic indicators that drive demand in this industry: prevailing interest rates, construction spending, production level, and consumer spending. Oil and gas prices specifically are also identified as drivers of demand because the level of energy exploration and extraction activities typically involve a heightened need for rented industrial equipment. On top of this, a rising energy price is usually indicative of overall growing economic activities across different sectors.

Another industry analysis report was published by Bluefield Research (2019) and depicts a recorded increasing trend of public spending in operations and maintenance of water utilities up to 2017 despite a decreasing trend in capital expenditure. The same report also illustrates the increasing number of weather-related disasters especially severe storms in recent years, to which the industry needs to respond. Reviews of these reports help us prioritize exogenous data to be collected and transformed into features in our machine learning models.

### 2.2 Machine Learning and Mathematical Optimization

The stochasticity of customer demand is the original and fundamental problem in supply chain management. As described in Introduction, the company has largely depended on the compartmentalized branch history when it comes to forecasting demand. In response, we propose utilizing a more robust data-driven method to improve forecast quality.

Various machine learning algorithms are increasingly used for numerical business-related prediction in a diverse set of industries. Neural networks, a subset of machine learning techniques, has especially enjoyed a revival of research interest in recent decades as more computing capability has become available (Hardesty, 2017). One of the earliest studies on machine learning applications for industrial demand forecasting was reported by Carbonneau et

al. (2008). They compared the forecast performance of machine learning algorithms, such as neural network and support vector machine, with the more conventional linear regression and other time-series methods, using real historical demand data from the Canadian steel foundry industry and dummy data from a supply chain simulation. For both datasets, the study showed that although machine learning approaches yielded an overall slightly superior forecast performance, the result did not differ significantly in a statistical sense from the result of the conventional techniques especially linear regression.

In the technology sector, Islam et al. (2012) constructed time-series predictive models using neural network and linear regression techniques for managing the allocation of computational resource in cloud computing industry. They also found that neural network algorithm returned better forecast accuracy than linear regression, especially when it was coupled with sliding-window technique (i.e., the training data used were a dynamic set of consecutive $n$ data points prior to data point $k$ and some prediction interval $r$). Meanwhile, in the energy sector, Maucec and Garni (2019) deployed machine learning algorithms to predict oil well production and highlight the importance of normalization to improve a model's predictive performance especially when it involves multiple variables with notably skewed distribution.

Additionally, considering the specific purpose and mostly emergency-related context of industrial pump rental market, the item level demand based on company data is noticeably intermittent. A conventional technique to forecast demand with such characteristics is based on Croston's method (Croston, 1972), which was subsequently revised based on a proposed correction over the estimated demand derivation by Syntetos & Boylan (2001). We also refer to a study by Lolli et al. (2017) where Croston's and Syntetos' methods are compared against a 2-layer neural network model for intermittent demand. Based on these works, we decided to approach the solution to the first part of our research problem (i.e., the predictive model) by exploring and comparing the forecasting performance of both conventional and machine learning techniques in the context of our sponsoring company's pump rental industry segment.

With a better-grounded demand prediction in place, we will have a clearer direction for how to improve the next stage of supply chain operations: distribution. As previously mentioned, the company's current asset allocation decision is mostly siloed among individual branches within each region. Equipment is moved from one branch location to another after the actual demand arrives. This firefighting mode of asset movement carries with it an inherent risk of lost rental

sales because of the time pressure to fulfil demand as well as of lost opportunity to optimize transportation cost. We propose, therefore, the integration of demand forecasting with a mathematical optimization model to improve allocation efficiency.

To the best of our knowledge, literature in which machine learning and optimization models are deployed consecutively as a unified system is not abundant. In a study by López Lázaro et al. (2018) both models are combined to improve the allocation of cash across automated teller machines (ATM) in the commercial banking industry. Asala et al. (2017) also deployed a similar model architecture to optimize the supply network of shale natural gas extraction. Both studies together cover the application of mixed linear, robust, and non-linear programming which use output from preceding machine learning models. Therefore, we conclude that this report can serve as a novel contribution to the literature on combined machine learning and mathematical optimization utilization, particularly in the context of industrial equipment rental industry.

## 3. METHODOLOGY

The flowchart in Figure 3-1 is a diagrammatic representation of all the major inputs, processes, and outputs which form the methodology of this capstone project. This methodology was developed in conjunction with the stakeholders from the sponsoring company. Based on a series of activities that included site visits and meetings, we clarified together the problem statement, scope, and the objective of the project. Subsequent steps in the project can be classified into 4 stages: Data Preparation, Predictive Modeling, Optimization Modeling, and Model Integration.

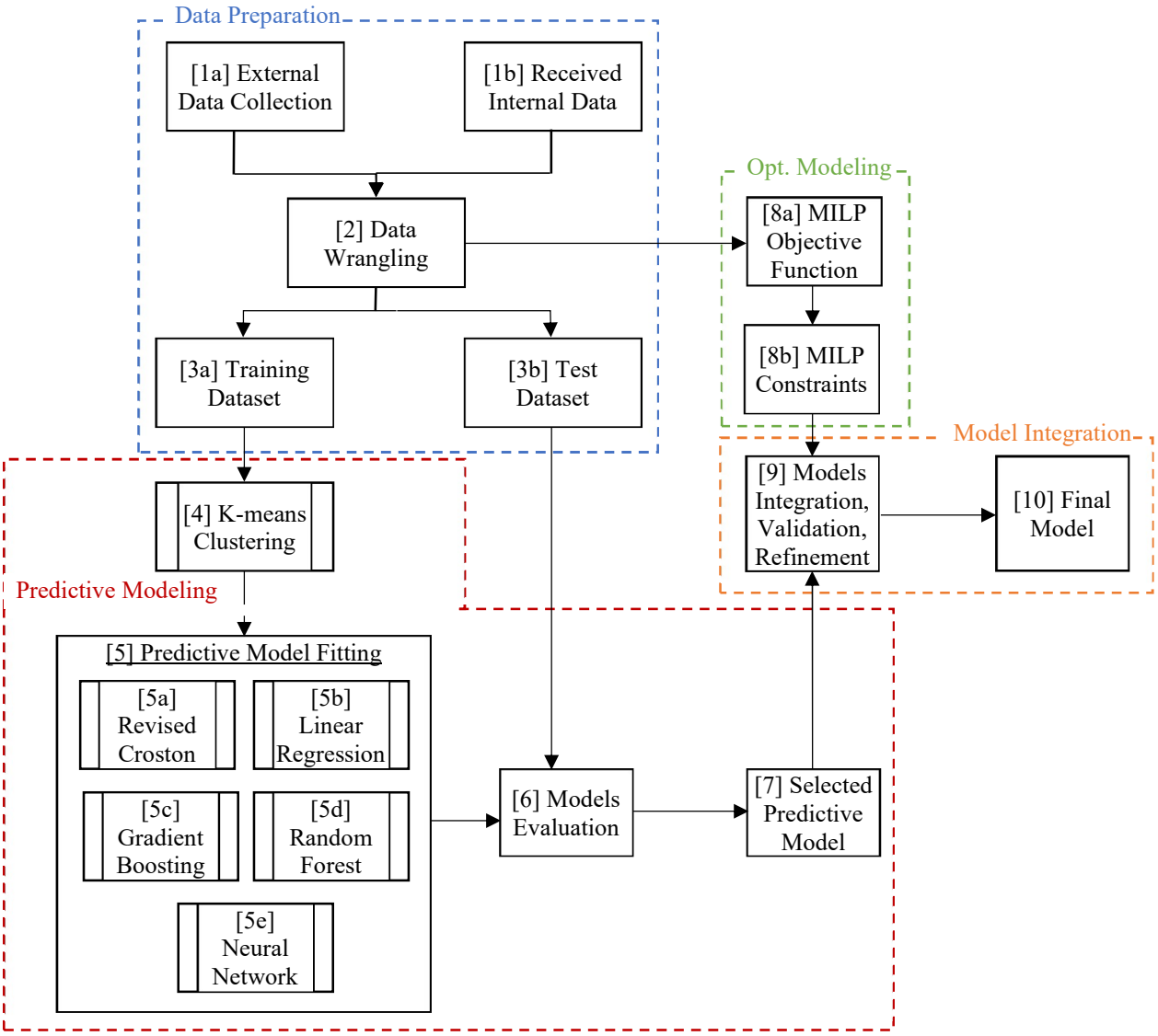


*Figure 3-1 Methodology Flowchart*

## 3.1. Data Preparation

In this section, we discuss about steps [1] to [3] in the methodology chart which concern the various aspects of the data that we received, collected, and used for predictive modeling purposes using both traditional and machine learning method. We start with the description and exploration of the main demand-related data that we received from the sponsoring company's internal system. Afterwards, we proceed with the description of various external data that we inferred to possibly have discernible correlation with the fluctuation of demand based on the literature review and inputs from the company's management. Finally, we discuss the way we extract and prepare trainable features from both internal and external data before feeding the train set into each of the predictive model algorithms.

### 3.1.1. Internal Data

Historical transaction data were available for extraction from the company's system and consolidated into a single dataset using Alteryx software. Alteryx is an analytics software platform used to reduce excessive time spent on data cleaning and preparation. The company was also able to provide us with an item code table, branch table, a project tracking report, and a detailed customer report. The item code table provides information about the characteristics of each item based on its item number, branch table specifies geographical position of each branch, project tracking report identified the market classification of each customer, and finally detailed customer report provided the billing and address information for every customer. An illustration of the internal data general structure can be seen in Figure 3-2.

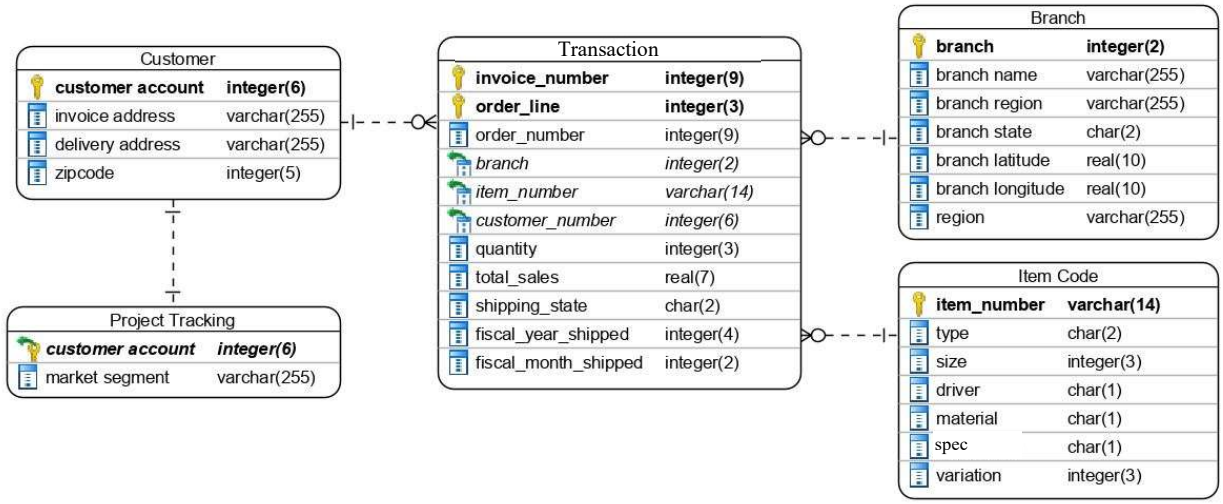


*Figure 3-2 Internal Data Relationship Diagram*

We used the field ‘Customer Number’ and ‘Item Number’, from historical transaction data, in a join operation with the field ‘Customer Account’, from customer report, and ‘Item Number’, from the item code table, to further consolidate all three tables into a single dataset. Before data exploration, we filtered the dataset to keep only relevant data. Out-of-scope transaction types which were not related to physical item rental activities, such as final sales and service, were excluded from the final dataset. We also eliminated negative sales and quantity values, which were primarily registered in the system only for credits or return transaction. As per the discussion with the company’s stakeholders, we further scoped down our dataset to focus on items under the main product family offered by the company across all of its branches. For machine learning modeling purpose, we split the resulting transaction data into 2 separate datasets. Data from the most recent year were set aside as the test set, it contains 61,578 rows or instances which would be used later for model evaluation. The rest of the data form 470,232 rows or instances and were designated as the training set, which would be used during machine learning modeling itself.

In order to gain a better understanding on the historical behavior of the demand we explored the dataset through perspective of demand value distribution, time series pattern, and geographical distribution. We observe first that the distribution of item demand values is heavily skewed to small positive integer around zero, as can be seen in Figure 3-3 where 95% of the values are less than or equal to 2. This pattern indicates an intermittent characteristic of the item’s demand overall.

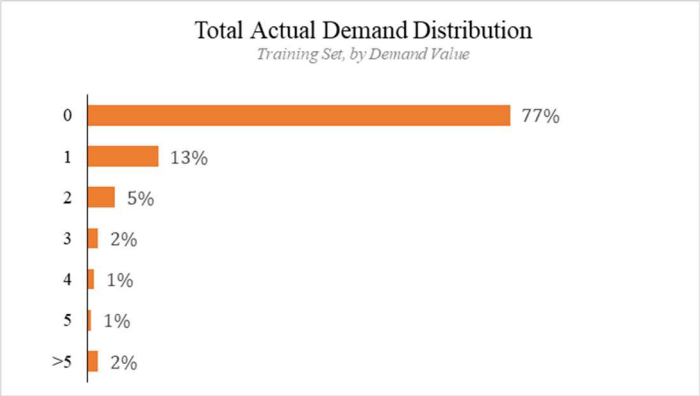


*Figure 3-3 Total Actual Demand Distribution*

We further investigate how demand behaves over a time series. As shown in Figure 3-4, rental demand seems to display quarterly seasonality in the form of a spike at the end of every

quarter. The highest demand among all months occurs in September, which is probably driven by the effects of hurricanes typically occurring during that time of the year (National Oceanic and Atmospheric Administration, 2020). Yearly, as displayed in Figure 3-5, demand seems to display a generally growing trend both since recorded data began.

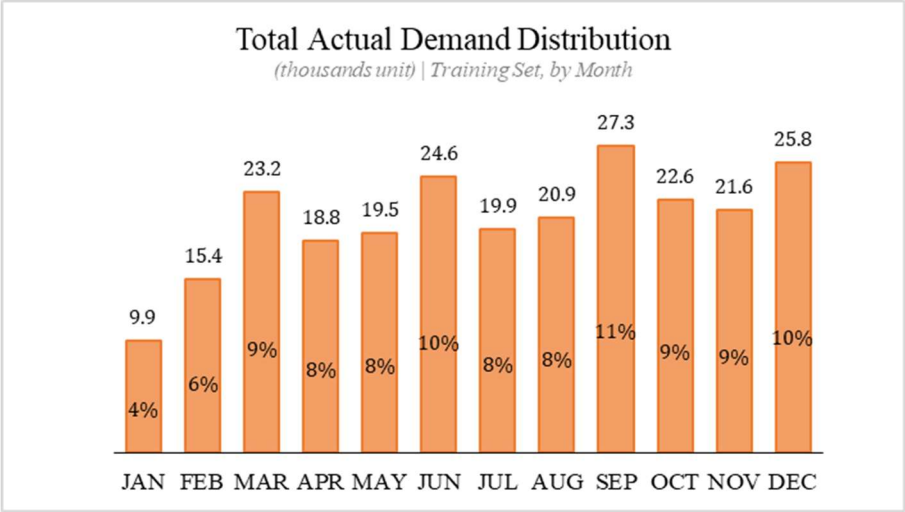


*Figure 3-4 Total Actual Demand Distribution by Month*

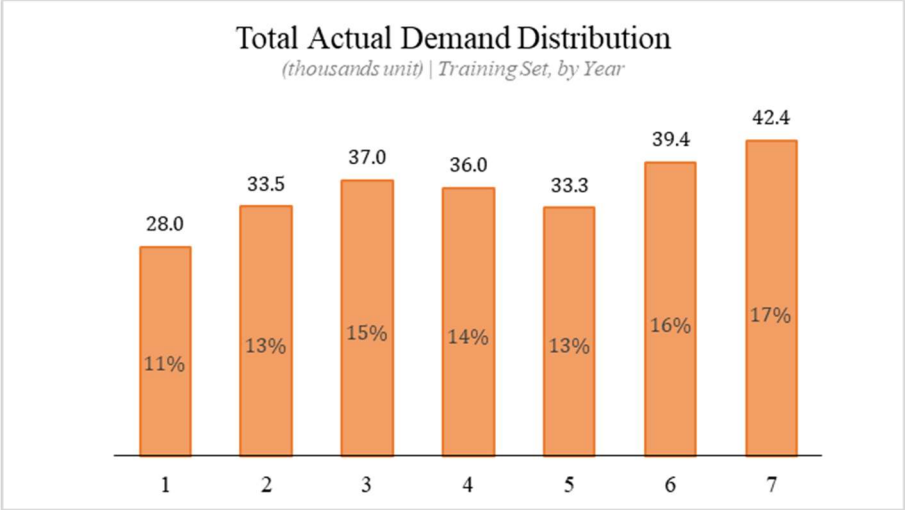


*Figure 3-5 Total Actual Demand Distribution by Year*

Geographically, total demand magnitude during the period covered in training set shows relatively high variability over the branches, as displayed by Figure 3-6. Some branches stood out with significantly higher demand than the others such as branch 1 and branch 19, followed by branch 15, 14, 9, 2 if we set an arbitrary threshold of total demand exceeding 10,000 items over the years just for exploratory discussion purpose. When we aggregate the demand by

regional clusters, as displayed in Figure 3-7, based on each branch's designated state location, which comprise the South, Northeast, Southeast, West, and Midwest, we see that overall more demand came from the both South and Southeastern part of the country, even though Northeast region on its own closely trails the Southern region.

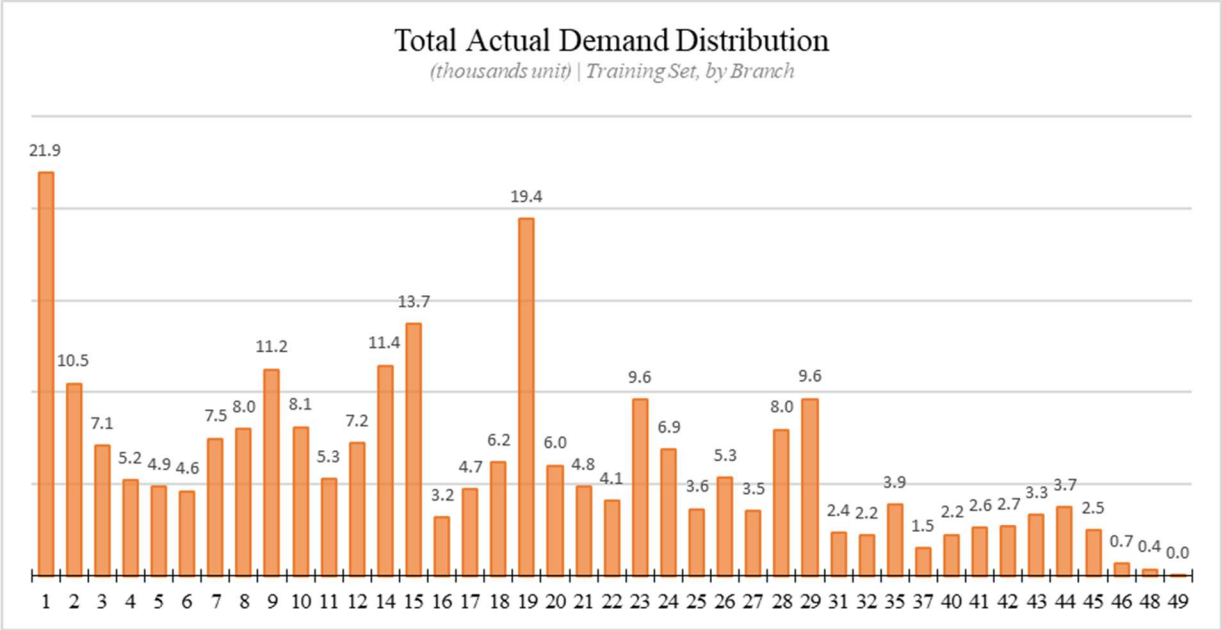


*Figure 3-6 Total Actual Demand Distribution by Branch*

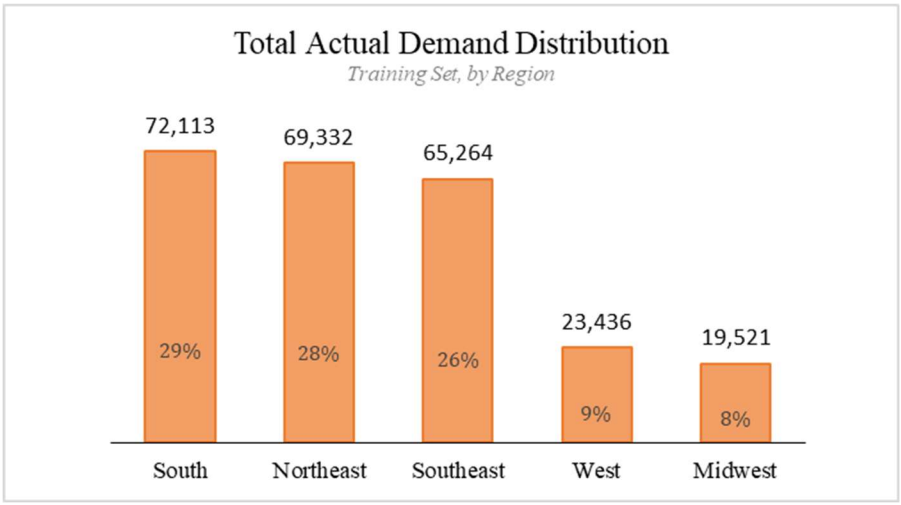


*Figure 3-7 Total Actual Demand Distribution by Region*

### 3.1.2. External Data

Table 3-1 lists down the external data we collected related to industrial activities from Federal Reserve Bank of St. Louis (2019), commodity futures prices from Ycharts (2019) and precipitation from National Centers for Environmental Information (NCEI) (2019). We collected

these data based on the insights we discuss on Literature Review section regarding the market drivers of industrial equipment industry where the sponsoring company operates.

*Table 3-1 List of External Data*

| External Data | Aggregation Level | Source |
|---|---|---|
| Industrial Production Index | National | FRED |
| Industrial Manufacturing | National | FRED |
| Nondurable Goods | National | FRED |
| Durable Manufacturing | National | FRED |
| New Orders - Non-Defense | National | FRED |
| Construction Spending - Sewage | National | FRED |
| Manufacturing New Orders | National | FRED |
| Mining Production Index (ex. Oil and Gas) | National | FRED |
| Total Industry Capacity Utilization Rate | National | FRED |
| Copper Futures | National | FRED |
| Natural Gas Futures Prices | National | Ycharts |
| Crude Oil Futures | National | Ycharts |
| Single Unit Housing Starts | National | FRED |
| ISM PMI Index | National | FRED |
| Rig Count | National | Ycharts |
| Capital Goods New Orders | National | FRED |
| Precipitation | State | NOAA |
| Precipitation Anomaly | State | NOAA |

We calculated the Pearson correlation (R-squared value) among the external data we have collected to investigate their potentials as exogenous features as displayed in Table 3-2. Some feature pairs indicate a phenomenon called multicollinearity, which occurs when a predictor variable is highly linearly related to one or more other predictor variables (here with a threshold of 0.8 R-squared value) and can lead to misleading results (Jacquillat, 2019). In order to prevent multicollinearity, we exclude Industrial Manufacturing index, Durable Goods Manufacturing, and Manufacturing New Orders from the predictive models.

*Table 3-2 Correlation Matrix of Potential Exogenous Features*

| Exogenous Features | | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 | F14 | F15 | F16 | F17 | F18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Industrial Production Index | F1 | | 0.9 | (0.6) | 0.9 | 0.3 | 0.3 | 0.8 | 0.7 | 0.7 | (0.6) | 0.6 | 0.4 | 0.7 | 0.4 | 0.6 | 0.2 | 0.1 | 0.1 |
| Industrial Manufacturing | F2 | 0.9 | | (0.6) | 0.8 | 0.4 | 0.5 | 0.7 | 0.6 | 0.5 | (0.5) | 0.5 | 0.3 | 0.8 | 0.4 | 0.4 | 0.2 | 0.1 | 0.1 |
| Nondurable Goods | F3 | (0.6) | (0.6) | | (0.8) | (0.0) | (0.2) | (0.4) | (0.5) | 0.0 | 0.6 | (0.1) | 0.2 | (0.7) | (0.3) | 0.0 | 0.1 | (0.1) | (0.1) |
| Durable_Manufacturing | F4 | 0.9 | 0.8 | (0.8) | | 0.2 | 0.3 | 0.7 | 0.7 | 0.3 | (0.7) | 0.4 | 0.1 | 0.8 | 0.4 | 0.4 | 0.1 | 0.1 | 0.1 |
| New Orders - Non-Defense | F5 | 0.3 | 0.4 | (0.0) | 0.2 | | 0.1 | 0.4 | 0.1 | 0.4 | (0.1) | 0.3 | 0.4 | 0.2 | (0.0) | 0.3 | 0.3 | 0.0 | 0.1 |
| Construction Spending - Sewage | F6 | 0.3 | 0.5 | (0.2) | 0.3 | 0.1 | | 0.2 | 0.4 | 0.1 | (0.3) | (0.0) | (0.1) | 0.5 | 0.1 | 0.0 | 0.0 | 0.1 | 0.1 |
| Manufacturing New Orders | F7 | 0.8 | 0.7 | (0.4) | 0.7 | 0.4 | 0.2 | | 0.4 | 0.6 | (0.3) | 0.5 | 0.4 | 0.6 | 0.5 | 0.5 | 0.7 | 0.1 | 0.1 |
| Mining Production Index (ex. Oil and Gas) | F8 | 0.7 | 0.6 | (0.5) | 0.7 | 0.1 | 0.4 | 0.4 | | 0.1 | (0.9) | 0.4 | 0.1 | 0.6 | (0.1) | 0.4 | (0.1) | 0.0 | 0.0 |
| Total Industry Capacity Utilization Rate | F9 | 0.7 | 0.5 | 0.0 | 0.3 | 0.4 | 0.1 | 0.6 | 0.1 | | 0.1 | 0.4 | 0.5 | 0.1 | 0.4 | 0.6 | 0.5 | 0.0 | 0.1 |
| Copper Futures | F10 | (0.6) | (0.5) | 0.6 | (0.7) | (0.1) | (0.3) | (0.3) | (0.9) | 0.1 | | (0.4) | (0.0) | (0.6) | 0.2 | (0.3) | 0.2 | (0.0) | (0.1) |
| Natural Gas Futures Prices | F11 | 0.6 | 0.5 | (0.1) | 0.4 | 0.3 | (0.0) | 0.5 | 0.4 | 0.4 | (0.4) | | 0.8 | 0.4 | 0.0 | 0.7 | 0.2 | 0.0 | 0.1 |
| Crude Oil Futures | F12 | 0.4 | 0.3 | 0.2 | 0.1 | 0.4 | (0.1) | 0.4 | 0.1 | 0.5 | (0.0) | 0.8 | | 0.1 | (0.0) | 0.5 | 0.4 | 0.0 | 0.0 |
| Single Unit Housing Starts | F13 | 0.7 | 0.8 | (0.7) | 0.8 | 0.2 | 0.5 | 0.6 | 0.6 | 0.1 | (0.6) | 0.4 | 0.1 | | 0.3 | 0.3 | 0.0 | 0.1 | 0.2 |
| ISM PMI Index | F14 | 0.4 | 0.4 | (0.3) | 0.4 | (0.0) | 0.1 | 0.5 | (0.1) | 0.4 | 0.2 | 0.0 | (0.0) | 0.3 | | 0.3 | 0.3 | 0.0 | 0.1 |
| Rig Count | F15 | 0.6 | 0.4 | 0.0 | 0.4 | 0.3 | 0.0 | 0.5 | 0.4 | 0.6 | (0.3) | 0.7 | 0.5 | 0.3 | 0.3 | | 0.3 | 0.0 | 0.0 |
| New Orders | F16 | 0.2 | 0.2 | 0.1 | 0.1 | 0.3 | 0.0 | 0.7 | (0.1) | 0.5 | 0.2 | 0.2 | 0.4 | 0.0 | 0.3 | 0.3 | | 0.0 | 0.1 |
| Precipitation | F17 | 0.1 | 0.1 | (0.1) | 0.1 | 0.0 | 0.1 | 0.1 | 0.0 | 0.0 | (0.0) | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 | | 0.6 |
| Precipitation Anomaly | F18 | 0.1 | 0.1 | (0.1) | 0.1 | 0.1 | 0.1 | 0.1 | 0.0 | 0.1 | (0.1) | 0.1 | 0.0 | 0.2 | 0.1 | 0.0 | 0.1 | 0.6 | |

### 3.1.3. Feature Engineering

Feature engineering is the process of transforming raw input data into input variables or features that the predictive machine-learning algorithm can interpret numerically. We deployed several methods such as aggregation and logarithm transformations, which we will explain further in this section. Figure 3-8 illustrates the summary of feature engineering steps in this project before modeling begins. These steps are adapted from the machine learning project checklist by Géron (2019).


*Figure 3-8 Diagram of Feature Engineering Steps*

In 'Aggregate' step, we ensured consistent aggregation levels for each feature. Internal demand data are available based on monthly invoice, so aggregation was monthly. Exogenous features were mostly already aggregated on monthly basis at the state level for precipitation and quarterly for some end-market related variables at national level which we interpolated to a monthly basis. This step leads to the determination of prediction period monthly at the item and branch level as an appropriate level of aggregation.

In 'Clean' step, we also scoped down the list of items to be included in the prediction model using the Pareto Principle where 20% of inputs are usually responsible for 80% of the output. In

this specific case, we found that during the most recent 3 years, 375 out of 3,383 items (i.e., around 10% of total unique items) accounted for 80% of the sales. Based on this observation, for the whole dataset which covers longer periods, we included 723 items in the demand model which make up approximately 90% of sales. Figure 3-9 illustrates the cumulative percentage of total sales quantity against the number of unique items for the Pareto analysis. Further exploration on demand magnitude also brought the total number of relevant branches included in the modeling to 41 branches.

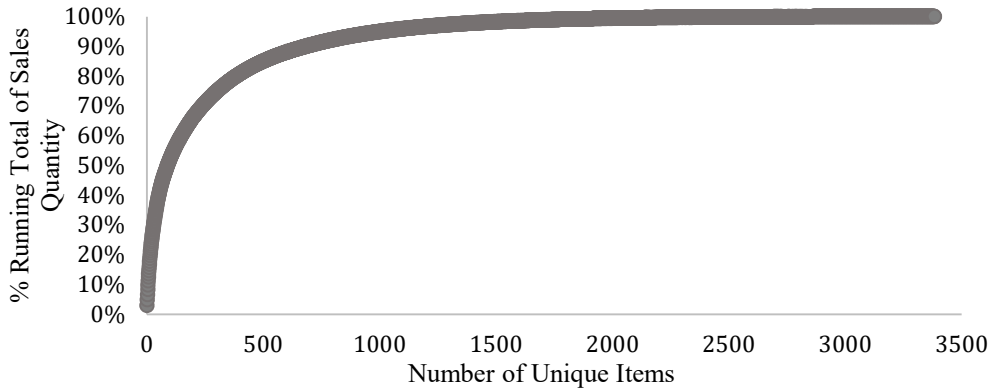


*Figure 3-9 Cumulative Percentage of Sales Quantity*

Machine learning algorithms cannot interpret categorical data, therefore, in step 'decompose' we transformed categorical variables into numerical variables using several encoding methods. Label encoding assign a unique numerical value to each class in the category. We used this technique in Linear Regression and Decision Tree-based models (Random Forest and XG-Boost). For example, the regions, instead of using Northeast, Southern, West, etc., are encoded as 1, 2, 3,…,7 under region column. We also applied a one-hot encoding method, whereby in a vector of binary values (0 or 1) of fixed length and order, according to the categorical class size and order, a category is represented by a value of 1 while the other vector element stays 0. For example, a category of value 2 within a category class of [1,2,3] is encoded as [0,1,0]. One-hot encoding is applied in the neural networks model for categorical data *Month* and the breakdown featurization of information in the item number.

For step 'transform', we implemented logarithmic transformation on the sales quantity data and created time series features. As shown in Figure 3-10, the distribution of sales quantity at the item-branch level are skewed toward the smaller values. Particularly, for linear regression,

random forest, and XG-boost models we did a log transformation on item-branch sales quantity. Logarithmic transformation helps to handle skewed data by creating a more normal distribution, and it also decreases the effect of outliers (see Figure 3-11) which depicts the spread of sales quantity after the logarithmic transformation. The range of the data is shorter, from 0 to 120 to 0 to 14, and the distribution is more evenly distributed across the range

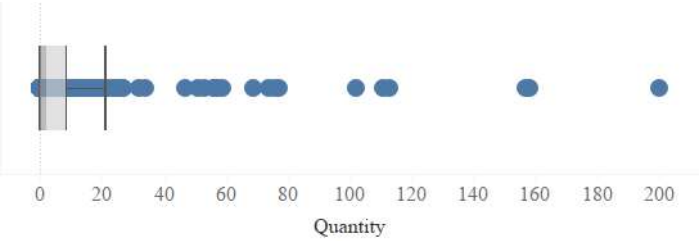


*Figure 3-10 Distribution of Sales Quantity before Log. Transformation*

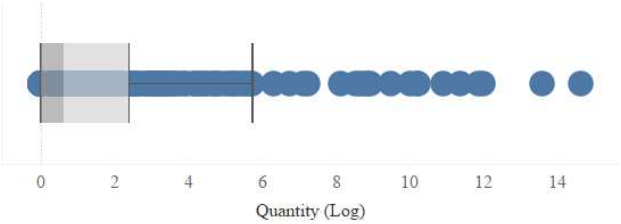


*Figure 3-11 Distribution of Sales Quantity after Log. Transformation*

To identify time-series features, like trend and seasonality, we created and tested a list of features as shown in Table 3-3.

*Table 3-3 Time-series Features*

| Time-series Features | Definition |
|---|---|
| Non_Zero_Demand_Intervals | Running count of consecutive periods of demand greater than zero |
| Cumulative_Zeros | Running count of consecutive periods of demand greater than zero |
| Avg_Sales_L3M | Average sales quantity for the last 3 months |
| Avg_Sales_L6M | Average sales quantity for the last 6 months |
| Avg_Sales_L9M | Average sales quantity for the last 9 months |
| Average_Sales_LTM | Average sales quantity for the last 10 months |
| P1Y_Month | Sales quantity at the same month in the previous year |
| P2Y_Month | Sales quantity at the same month in the previous 2 years |
| Previous_Month | Sales quantity in previous month |
| Previous_2Month | Sales quantity in previous 2 months |
| Previous_3Month | Sales quantity in previous 3 months |
| YoY_Change_Avg_Sales_LTM | Ratio between average sales quantity for the last 10 months at current year relative to last year |

Lastly, in the 'Scaling' step, we map features with numerical values to different scales. For the K-means clustering, we normalized features with numerical values relative to their minimum

and maximum range (i.e., MinMax scaling). This estimator scales each feature individually, so the given range is between zero and one. In neural networks model, data are scaled for feature *size* and *variations* which are part of the breakdown featurization of information contained in the item number, and feature *lat* and *long* which represent the latitude and longitude of branches and are used as features instead of branch number. For all other numerical features in the neural networks model, the standardization or standard scaling is used. We refer to Géron (2019) for both of these scaling methods.

### 3.2. Predictive Modeling

In this stage, we enter the modeling part (step [4] to [7] in the Methodology flowchart, Figure 3-1) of the project starting with K-means clustering which is a form of unsupervised learning. As an unsupervised learning algorithm, in K-means clustering there is not a dependent variable to predict (Géron, 2019). The aim is instead to create groups of similar data points only for exploratory purpose. For demand prediction, we started with using Revised Croston's method as the conventional forecasting method for intermittent data. Subsequently, we used Python to deploy several supervised machine learning algorithms, where historical target value in the training data exist. The machine learning algorithms involved are: Least Square Linear Regression, Decision Tree Models (Random Forest and Gradient Boosting), and Feed-forward Neural Networks. In comparing the performance of the predictive models, we chose to use Mean Squared Error (MSE) which is defined as follows,

$$MSE = \frac{\sum_{i=1}^{n}\left(\delta_i - \hat{\delta}_i\right)^2}{n} \quad (1)$$

where $n$ represents the total number of data points, $\delta_i$ represents actual demand, and $\hat{\delta}_i$ represents the predicted demand. MSE formulation allows us to penalize large error, due to the squaring, while permitting small error values to avoid training set overfitting during the modeling. We did not include Mean Average Percentage Error (MAPE) as performance criteria due to the sparse actual demand values. As displayed in Figure 3-3, over 70% of data points have actual demand value of 0 (zero), in practice this means MAPE can only be calculated for less than 30% of the historical data points, which is obviously not representative of the whole demand pattern. For additional external reference, Géron (2019), for example, also uses MSE in various implementation of regression models in his work.

### 3.2.1. Revised Croston's Method

Based on our observation during data preparation, demand pattern seems to be intermittent. We therefore choose to start with Croston's time-series forecast method for the conventional approach. We refer to the exposition by Silver et al. (2016b) in operationalizing the original Croston's method with revision proposed by Syntetos & Boylan (2001). Forecasted demand for period *t+1* or $\hat{\delta}^{(t+1)}$ estimated at the end of period *t* is defined as follows,

$$\hat{\delta}^{(t+1)} = \frac{\hat{z}^{(t)}}{\hat{n}^{(t)} c^{\hat{n}^{(t)}-1}} \tag{2}$$

where $\hat{z}^{(t)}$ is the average demand size estimated at the end of period *t*, $\hat{n}^{(t)}$ is the estimated period interval from the most recent non-zero demand , and *c* is a constant we set to be equal to 100 in this implementation. In addition, $\hat{z}^{(t)}$ and $\hat{n}^{(t)}$ are updated with a smoothing factor $\alpha$ where $0 \leq \alpha \leq 1$ each time actual non-zero demand occurs:

$$\hat{z}^{(t)} = \alpha\delta^{(t)} - (1-\alpha)\hat{z}^{(t-1)} \tag{3}$$

$$\hat{n}^{(t)} = \alpha n^{(t)} - (1-\alpha)\hat{n}^{(t-1)} \tag{4}$$

where $\delta$ and $n$ is actual demand and period interval from the most recent non-zero demand.

### 3.2.2. Least Squares Linear Regression

An interpretable and straightforward model is the least-square linear regression. This method works by finding a set of coefficient values $(B_1)$ for the independent variable $(x_i)$ and offset $B_0$ which minimizes the square error $(e_i)$ of the actual historical values of the dependent variable $(y_i)$ against the prediction results, as represented in equation (5). Linear regression is used primarily in a situation when there is inability to access nonlinear relationships (Lee & Shin, 2020).

$$y_i = B_0 + B_1 x_i + e_i \tag{5}$$

### 3.2.3. Decision Tree Models (Random Forest and Gradient Boosting)

Random Forest and Gradient Boosting are a regression and classification tree algorithms. Decision trees create maps of questions about predictor variables to determine the final output. The decision tree model in this project is built and trained using the Scikit-learn library in Python which uses what is called Classification and Regression Tree (CART) algorithm (Géron, 2019). As such, the model comprises a series of nodes, each of which has a threshold value and

splitting binary branches, and leaves which are terminal nodes with no further branching. The first node is called the root node which then branches out to the subsequent depth or level with leaves or another branching node, which is termed child node or internal node. Figure 3-12 illustrates the structure of a decision tree model for clarity of model representation. Naturally for demand prediction purpose, the model is designed to fulfil regression task

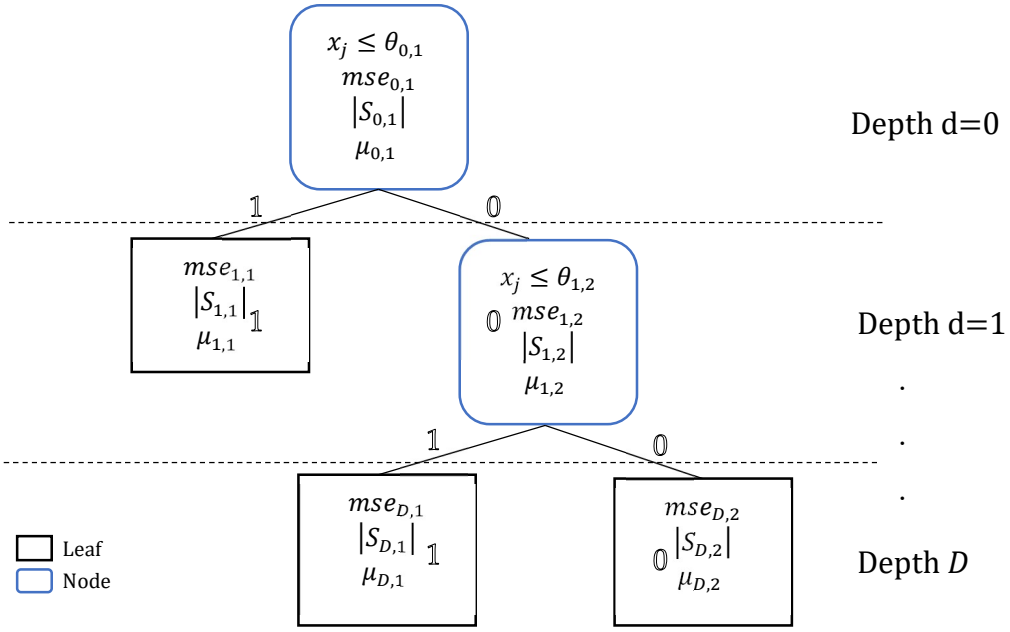


*Figure 3-12 Decision Tree Structure Illustration*

Let us define the set of training samples $\{(x^{(i)}, y^{(i)}), i = 1, \dots, t\}$ with input feature vectors $x \in \mathbb{R}^n$ and targets $y \in \mathbb{Z}_{\geq 0}$ (non-negative integers). We build a decision tree where at each node, a set of training samples $S_{d,m}$ is split into 2 subsets at the subsequent depth based on the splitting rule $\mathbb{1}_{[x_j \leq \theta_{d,m}]}$ where $j \in \{1, \dots, n\}$ is an index of an element of input feature vector $x$ used as predictor and $\theta_{d,m} \in \mathbb{R}$ is a threshold value determined for $x_j$ at node $m$ on depth $d$. If we refer to the decision tree illustrated in Figure 3-12, at the root node for example, each training instance $i$ is moved to the leaf on the left at depth 1 for all $i$ where $x_j^{(i)} \leq \theta_{0,1}$ is true ($\mathbb{1}$) and to the child node on the right for all $i$ where $x_j^{(i)} \leq \theta_{0,1}$ is false ($\mathbb{0}$). Note that within and across depth, a different $j$ might be used by different nodes. The split then continues down to a certain depth D which contains only leaves.

There are some notable attributes associated with the set at each $node_{d,m}$ which contribute to the way prediction is optimized. $|S|$ represents the number of samples in the set, $\mu$ represents the

average value of $y$ (i.e., $\sum_{y^{(i)} \in S} \frac{y^{(i)}}{|S|}$) and finally $MSE$ is simply the mean square error as formulated in equation (1) where in this case $\delta_i = y^{(i)}$, $\hat{\delta}_i = \mu$, and $n = |S|$. For every leaf node, $\mu$ therefore represents the predicted value for each input instance $x^{(i)}$ that traverses the tree and eventually ends up there.

The CART algorithm trains the decision tree model by looking for a pair of $x_j$ and $\theta$ at each node such that the loss function $J(x_j, \theta)$ is minimized. If we define notational index for the child nodes of a $node_{d,m}$ as $node_{d,m}^{\mathbb{1}}$, which contains all instances $i$ where $x_j^{(i)} \leq \theta_{d,m}$ is true, and $node_{d,m}^{\mathbb{0}}$ for all $i$ where $x_j^{(i)} \leq \theta_{d,m}$ is false, then the loss function is defined as follows,

$$J(x_j, \theta_{d,m}) = \frac{|S_{d,m}^{\mathbb{1}}|}{|S_{d,m}|} mse_{d,m}^{\mathbb{1}} + \frac{|S_{d,m}^{\mathbb{0}}|}{|S_{d,m}|} mse_{d,m}^{\mathbb{0}} \quad (6)$$

In general, better prediction performance is achieved through an aggregate or ensemble of multiple prediction models instead of just one (Géron, 2019). We therefore implement two forms of ensemble decision tree models in this project: Random Forests and Gradient Boosting. In Random Forest, 50 decision trees are initially involved in training a subset of randomly drawn training samples and input features which differs for each tree. The final prediction is simply the mean of all values predicted by the trees. During training for Random Forest, we also tune the following regularization hyperparameters to search for the optimal architecture within the specified search space:

- Maximum number of features subset : (3,8]
- Maximum depth (D) : (6,11]
- Minimum $|S|$ in a node before split : (5,15]
- Minimum $|S|$ in a leaf : (5,15]
- Bootstrap : {0,1}

Note that hyperparameter 'bootstrap' specifies whether random sampling of the training set is done with replacement when Bootstrap is True or without replacement otherwise.

In contrast to Random Forest where multiple decision trees are trained in parallel, Gradient Boosting is an iterative process and does not aggregate multiple models. The trees are sequential models and do not complement each other. The algorithm draws from random training samples, uses them to fit a tree, subtracts the prediction from the original data, and creates a new tree (Nawar & Mouazen, 2017). The explanation for gradient boosting algorithm is as follows,

$$1 - Initialize\ f_0(x) = \arg min_\lambda \sum_{i=1}^{N} L\ (y_{i,}\ \lambda) \tag{7}$$

For *m*=1 to *M*:

a) For i = 1, 2…, *N* compute

$$r_{im} = -\left[\frac{\partial L\left(y_{i'} f(x_i)\right)}{\partial f(x_i)}\right] \tag{8}$$

b) Fit a regression tree to the targets $R_{im}$

$$R_{im}, j = 1,2 \ldots . J_m \tag{9}$$

c) For j = 1, 2…, $J_m$ compute

$$\lambda_{jm} = \arg min_\lambda \sum_{i=1}^{N} L\ (y_{i,}\ f_{m-1}(x_i) +\ \lambda\ ) \tag{10}$$

d)

$$\text{Update } f_m(x) = \left(\ f_{m-1}(x) + \sum_j \lambda_{jm}\ \right). \tag{11}$$

The first line (a) of the algorithm initializes the optimal constant model, which is just single terminal node tree. The components of the negative gradient computed at line (b) are the residuals *r*. Learning rate $\lambda$ parameterizes the spilt variables and spilt points at the terminal nodes in (c). At each iteration *M,* the algorithm solves for the optimal loss function and adds to the current expansion current expansion $f_{m-1}(x)$ in (d). Standard parameters are the number of iterations *N,* the number of trees *M*, and the sizes of each of the trees $J_m$.

### 3.2.4. Feed-forward Neural Networks

We built a feed-forward neural networks (FNN) with a single hidden layer which is proven to yield superior forecast performance for intermittent demand compared to traditional method in a study by Lolli et al. (2017). Since demand forecasting is a regression task, we only need one neuron with rectified linear unit (ReLU) as the activation function at the output layer. ReLU function outputs any non-negative input as is and zero for the negative one. This makes sure that

the output of the model will always be at least zero. For visual reference, a graphical overview of the FNN architecture is displayed in Figure 3-13.

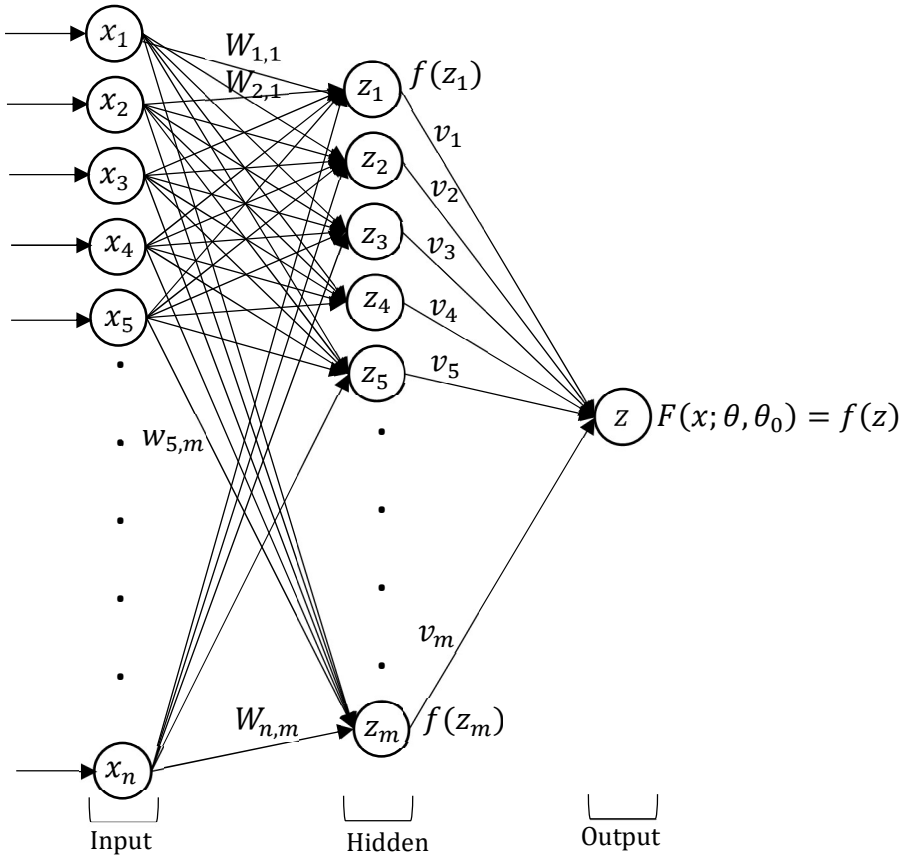


*Figure 3-13 Feed-Forward Neural Network Architecture*

Géron (2019) specifies typical hyperparameter architecture for regression FNN which we use in this model including number of neurons per hidden layer (at least 10), activation function for hidden unit (ReLU), loss function (MSE), as well as learning rate and regularization parameter which will be explained later. We also refer to the same literature for practical coding approaches using NumPy, Pandas, Scikit-learn, and TensorFlow-Keras libraries in Python. All the programming was run in a laptop with 16GB of RAM and Windows 10 Home (64-bit) operating system. The model's mathematical representation, which is discussed in the following paragraphs, is an adaptation of FNN backpropagation explanation in the lecture notes of Jaakkola & Barzilay (2016) of the Massachusetts Institute of Technology (MIT).

Let us define again this time a given set of $t$ training samples $\{(x^{(i)}, y^{(i)}), i = 1, \dots, t\}$ with input vectors $x \in \mathbb{R}^n$ and targets $y \in \mathbb{Z}_{\geq 0}$ (non-negative integers). For each input $x_{raw}^{(i)}$ in the raw training samples, we apply a featurization function $\phi$ such that $\phi\left(x_{raw}^{(i)}\right) = x^{(i)} = [x_1^{(i)}, \dots, x_n^{(i)}]$ which corresponds to the neurons in the input layer.

In the hidden layer, the output of each of the $m$ neurons is calculated with 2 steps represented in equations (12) and (13). In equation (12) $W_{j,k}$ represents the multiplier weight of input neuron $x_j$ to hidden neuron $z_k$; while $W_{0k}$ represents the bias parameter. In equation (13) the resulting weighted sum of $n$ input neurons is then passed through the ReLU function $f(z_k)$.

$$z_k = \sum_j^n x_j W_{j,k} + W_{0k} \tag{12}$$

$$f(z_k) = \max\{0, z_k\} \tag{13}$$

In the output layer, the single neuron $z$, in turn, takes in the weighted sum of $m$ hidden nodes activation $f(z_k)$. This is represented by equation (14), where $V_k$ is the weight for each hidden neuron $k$ and $V_0$ is the scalar bias. The value of $z$ is then passed through the ReLU function $F(x; \theta, \theta_0)$ in equation (15) for the final prediction output. Moreover, $\theta$ and $\theta_0$ represent the trainable parameters in the model where $\theta = \{W_{j,k}, V_k\}$ and $\theta_0 = \{W_{0k}, V_0\}$.

$$z = \sum_k^m f(z_k) V_k + V_0 \tag{14}$$

$$F(x; \theta, \theta_0) = \max\{0, z\} \tag{15}$$

The model is then trained to find a set of parameters $\theta$ such that the following average loss function over the training samples is minimized:

$$J(\theta) = \frac{1}{t}\sum_i^t Loss\left(y^{(i)} F\left(x^{(i)}; \theta\right)\right) = \frac{1}{t}\sum_i^t \left[\left(y^{(i)} - F\left(x^{(i)}; \theta\right)\right)^2 + \frac{\lambda}{2}\|\theta\|^2\right] \tag{16}$$

Note that $\frac{\lambda}{2}\|\theta\|^2$ is the ridged regression-based regularization term added to penalize large weight magnitude (Géron, 2019) where $\lambda \in \mathbb{R}_{>0}$. The lost function for $\theta_0$ is similar but without the regularization term.

We chose stochastic gradient descent (SGD) algorithm for loss minimization due to its faster computational time especially when the training set grows larger over time (Géron, 2019). We begin SGD by finding the partial derivatives of the loss function with respect to the parameters

for each of the randomly ordered training sample$(x^{(s)}, y^{(s)})$, as shown in equations for weights (18) - (21). Note that we use a representation of Iverson bracket where:

$$[\![\cdot]\!] = \begin{cases} 1, if\ True \\ 0, if\ False \end{cases} \tag{17}$$

$$\frac{\partial}{\partial W_{j,k}} Loss\left(y^{(s)} F(x^{(s)};\theta)\right) = \left[\frac{\partial z_k^{(s)}}{\partial W_{j,k}}\right]\left[\frac{\partial f(z_k^{(s)})}{\partial z_k^{(s)}}\right]\left[\frac{\partial z^{(s)}}{\partial f(z_k^{(s)})}\right]\left[\frac{\partial F(x^{(s)};\theta)}{\partial z^{(s)}}\right]\left[\frac{\partial}{\partial F(x^{(s)};\theta)} Loss\left(y^{(s)} F(x^{(s)};\theta)\right)\right] \tag{18}$$

$$= [x_j][\![z_k^{(s)} > 0]\!][V_k][\![z^{(s)} > 0]\!]\left[2(F(x^{(s)};\theta) - y^{(s)})\right] + \lambda W_{j,k} \tag{19}$$

$$\frac{\partial}{\partial V_k} Loss\left(y^{(s)} F(x^{(s)};\theta)\right) = \left[\frac{\partial z^{(s)}}{\partial V_k^{(s)}}\right]\left[\frac{\partial F(x^{(s)};\theta)}{\partial z^{(s)}}\right]\left[\frac{\partial}{\partial F(x^{(s)};\theta)} Loss\left(y^{(s)} F(x^{(s)};\theta)\right)\right] \tag{20}$$

$$= \left[f\left(z_k^{(s)}\right)\right][\![z^{(s)} > 0]\!]\left[2(F(x^{(s)};\theta) - y^{(s)})\right] + \lambda V_k \tag{21}$$

Each parameter is then updated with the opposite "direction" of the derived gradient, with $\eta \in \mathbb{R}_{>0}$ representing the learning rate, as represented in equation (22) and (23).

$$W_{j,k} \leftarrow W_{j,k} - \eta([x_j][\![z_k^{(s)} > 0]\!][V_k][\![z^{(s)} > 0]\!]\left[2(F(x^{(s)};\theta) - y^{(s)})\right] + \lambda W_{j,k}) \tag{22}$$

$$V_k \leftarrow V_k - \eta\left(\left[f\left(z_k^{(s)}\right)\right][\![z^{(s)} > 0]\!]\left[2(F(x^{(s)};\theta) - y^{(s)})\right] + \lambda V_k\right) \tag{23}$$

Based on the model formulation, we also applied grid search approach using Scikit-learn with 5-fold cross validation to tune the following hyperparameters in a search for a more optimum architecture:

- Number of hidden neurons : {10,11,12}
- Learning rate ($\eta$) : {0.1, 0.01, 0.001}
- Regularization parameter ($\lambda$) : {0.01, 0.001, 0.0001}

Finally, we analyzed feature importance of the resulting model using connection weight method as is operationalized in a study by Olden & Jackson (2002) and showed to be more accurate and explanatory than the more widely used Garson's method (Olden et al., 2004). In the

connection weight approach, the importance of each feature $j$ or $Ix_j$ is simply represented as follows:

$$Ix_j = \sum_{k=1}^{m} W_{j,k} V_k \tag{24}$$

3.3. Mathematical Optimization Modeling

This stage covers steps [8a] and [8b] in the methodology flowchart where we develop a solution to optimize the second problem of this capstone project which concerns the company's asset movement. Despite their numbering, the steps in this stage were performed in parallel with the Predictive Modeling stage because the structure of predictive model output had been defined at the conclusion of the Data Preparation stage. For building and running the model, we used a commercial optimization solver engine Gurobi Optimizer version 9.0.0 build v9.0.0rc2 on Python 3 interface and web-based Jupyter Notebook platform. The solver was run in the same computer hardware that we used for modeling FNN.

The following sub sections contain several mathematical formulations written in the following notational style and convention. We use uppercase Latin letters to denote major elements of the objective function and lowercase ones for parameters and the decision variable. Latin letters written in subscript represent categorical indices or class, while we place the chronological time period index inside brackets in superscript. We use Greek letters to denote any external input parameters that affect the value of parameters in the optimization model. The main and recurring symbols are as follow:

- $t$: time period index (month); $\mathrm{t} \in \{\mathrm{t}, \mathrm{t}+1, \ldots, \mathrm{n}\}$
- $k$: item index; $\mathrm{k} \in \{0,1, \ldots, \mathrm{K}\}$; $\mathrm{K} = 723$
- $i$: origin branch index; $i \in \{0,1, \ldots, \mathrm{I}\}$; $\mathrm{I} = 47$, including dummy origin $i = 0$
- $j$: destination branch index; $j \in \{1, \ldots, \mathrm{J}\}$; $\mathrm{J} = 46$

Based on this convention, $x_{kij}^{(t)}$ denotes the number of pumps type $k$ moved from branch $i$ to branch $k$, at time period $t$; and represents the decision variables in our mathematical optimization model.

### 3.3.1. Objective Function

As with any optimization model, we started with [8a] Objective Function Formulation. Given the context of this stage, the objective function is defined as the total cost incurred in the allocation of pumps among the company's branches within a planning period. Expert consensus has identified the types of costs which are relevant for inventory management problem as unit variable cost, inventory carrying cost, ordering or setup cost, and stockout cost (Silver et al., 2016a). Within the scope of optimization model in this project, transportation cost is the most relevant variable cost. We therefore define the objective function of our optimization model as follows:

$$\min \sum_{t} \sum_{k} \sum_{i} \sum_{j} c_{kij}^{(t)} \, x_{kij}^{(t)} \tag{25}$$

where $c_{kij}^{(t)}$ is the variable transportation cost for each pump item $k$ transported from branch $i$ to branch $k$ at time period $t$. Transportation cost of item from dummy origin branch 0 to any destination branch will be artificially set to be always the most expensive of all origins to make the model utilize supply from the branch only as the last resort. Although intuitively we use values in dollar term for this parameter, based on data availability and the company's preference to compare year-to-year allocation performance from a purely operational efficiency perspective, it was decided to use distance (miles) between branches as the variable transportation cost. We collected estimate latitude ($LAT$) and longitude ($LON$) in radian for each branch location and calculated the approximate distance of each pair of branch ($D_{ij}$) in miles using the great-circle distance formula:

$$D_{ij} = 3959 \arccos\left(\sin LAT_i \sin LAT_j + \cos LAT_i \cos LAT_j \cos\left(LON_i - LON_j\right)\right) \tag{26}$$

### 3.3.2. Constraints

The constraints are defined as follows,

$$s_{k0}^{(t)} = \max\left(0, \sum_{j} \sum_{i} d_{kj}^{(t)} - s_{ki}^{(t)}\right), \forall \, i > 0, k, t \tag{27}$$

$$\sum_{i} x_{kij}^{(t)} \geq d_{kj}^{(t)}, \forall \, k, j, t \tag{28}$$

$$\sum_{j} x_{kij}^{(t)} \leq s_{ki}^{(t)}, \forall\, k, i, t \tag{29}$$

$$\sum_{k} x_{kij}^{(t)} \leq l_{ij}^{(t)}, \forall\, i, j, t \tag{30}$$

$$x_{kij}^{(t)} \in \mathbb{Z}_{\geq 0}, \forall\, i, j, t \tag{31}$$

Constraint (27) specifies an artificial item supply quantity assigned to dummy origin branch 0 in the case of total item net demand exceeding total item net supply, where the value is the difference between the two. This enables the model to be always computationally feasible as well as allowing the stockout distribution to be interpreted by looking at all items transported from dummy origin branch 0 in the result.

Constraint (28) restricts the number of certain pump *k* allocated to each destination branch *j* to be at least equal to the net demand $d_{kj}^{(t)}$ which is calculated at the beginning of period *t* according to the following step:

$$d_{kj}^{(t)} = \max\left(0, \hat{\delta}_{kj}^{(t)} - \xi_{kj}^{(t-1)}\right) \tag{32}$$

$\hat{\delta}_{kj}^{(t)}$represents the predicted demand for a period while $\xi_{kj}^{(t-1)}$ denotes the level of stock as recorded at the end of the preceding period. In effect, the formula returns 0 if predicted demand is lower than the recorded stock since it means that there is no need to transport any additional item to that branch. The predicted demand parameter comes from the output of predictive model while the item stock level parameter comes from the actual stock record which is updated at the end of each planning period.

Constraint (29), on the other hand, restricts the number of item which is sent out of an origin branch to be no more than the net supply parameter $s_{ki}^{(t)}$. As with net demand, this parameter is also calculated using external input parameters, as follows:

$$s_{ki}^{(t)} = \max\left(0, \xi_{kj}^{(t-1)} - \hat{\delta}_{kj}^{(t)}\right), \forall\, i > 0 \tag{33}$$

which means that an item is allowed to be transported out of a branch only when the predicted demand is lower than the recorded stock level.

Constraint (30) enforces a limit on the number of transported items on each lane ($l_{ij}^{(t)}$). Finally, constraint (31) requires the decision variable to be non-negative integer.

## 3.4. Model Integration

After the predictive models were evaluated and the best performing one is selected, it was then integrated with the MILP model as a unified decision tool (step [9] and [10] in the methodology flowchart). As both models were developed using the same Python programming language, they can be easily integrated in one platform which, for simplicity and codes accessibility, takes form of a Jupyter notebook. Figure 3-14 illustrates the working diagram of how the integrated decision tool operates. The selected prediction model will automatically feed demand prediction result to the MILP model, with a possibility of intervention for manual adjustment if deemed appropriate. The subsequent MILP model takes in additional data as parameters to run the optimization iterations which consist of Lane Capacity, Transportation Cost, and Item Stock per Branch. The final output of the whole tool is a table prescribing the most efficient asset allocation under the given parameters. Consistent with the forecast period in the predictive model, we design this tool to be used in a monthly basis. We attach a more technical user guide to run and maintain this tool in Appendix A.

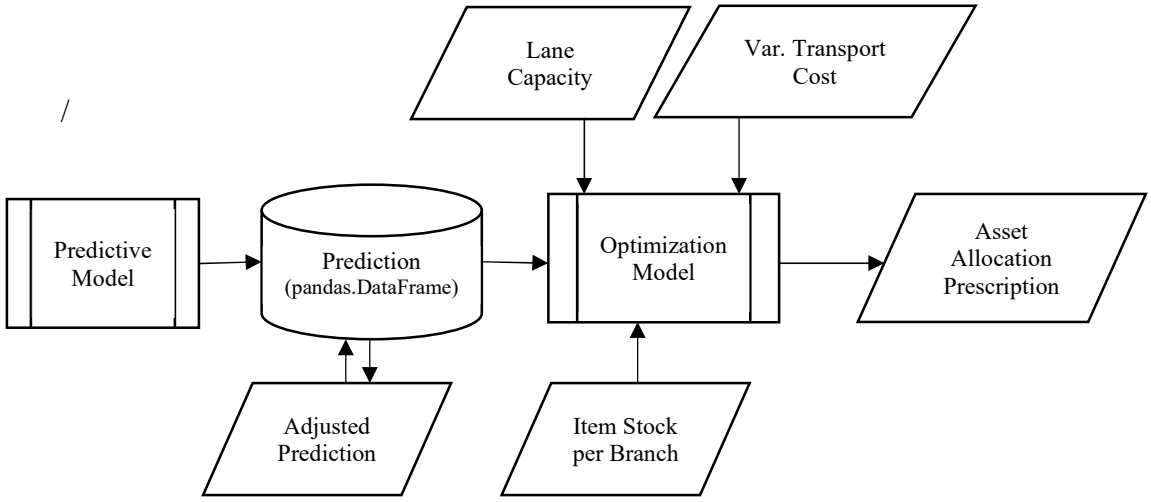


*Figure 3-14 Working Diagram of Integrated Decision Tool*

## 4. RESULTS AND DISCUSSION

In this section we discuss the results of predictive and mathematical optimization models as discussed in Methodology.

### 4.1. Clustering

Each item was categorized into one four categories for a total of 639 items based on year over year change in sales, percentage of sales by industry, and percentage of sales in each region. We determined clustering attributes from conversations with executives at the Company. Demand profiles are segregated based on geography and industry application. We included changes in sales for the past three years to improve the identification of trends in the forecasting algorithms. To determine the number of clusters, we used the elbow method (Kuraria et al., 2018) with result displayed in Figure 4-1. Within-Cluster-Sum-of Squares (WCSS) increases with the number of clusters; however, the rate at which WCSS's increases drops after a certain number of clusters. By selecting the number of clusters linked to the elbow point, the groups are more compact and meaningful. Using this approach, we decided on four clusters. Table 4-1 shows the results of the clustering. Each geographic region influences a cluster. For example, Cluster 2 consists of the majority of items with sales in the West Region, while Cluster 3 consists of the Northeast Region. We incorporated clusters as a feature in the initial forecasting algorithms of decision tree-based models; however, in the final model, we excluded clusters due to their insignificance as they were ranked in the bottom quartile among all features.

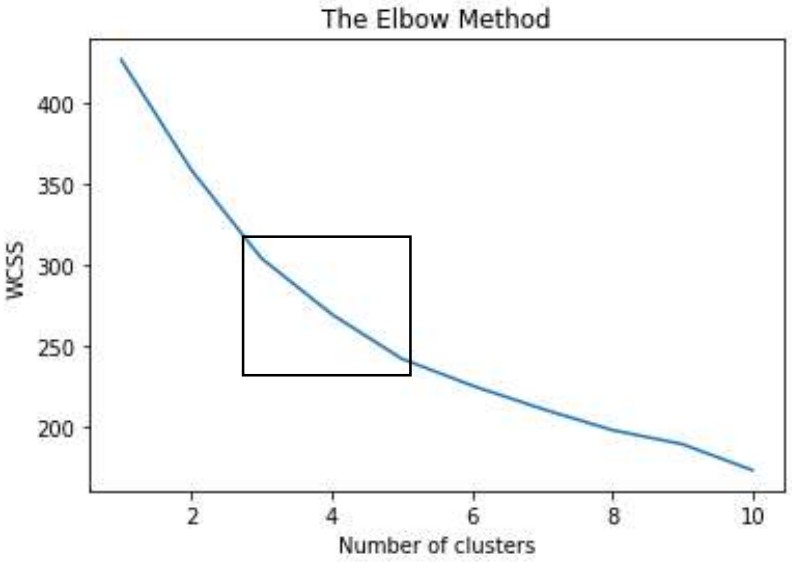


*Figure 4-1 Clustering Elbow Method Result*

*Table 4-1 Clustering Result*

| Attributes | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|---|
| Number of Items | 158 | 46 | 192 | 240 |
| Changes in Sales: Year 1-2 | 2,300,788 | (255,568) | 1,281,701 | 1,866,857 |
| Changes in Sales: Year 2-3 | 2,694,295 | 2,559 | 2,843,932 | 2,269,887 |
| Changes in Sales: Year 3-4 | 23,132 | 222,329 | 1,292,079 | 2,315,172 |
| % of sales in Construction | 40% | 42% | 40% | 43% |
| % of sales in Industry 1 | 9% | 4% | 7% | 7% |
| % of sales in Industry 2 | 16% | 12% | 6% | 16% |
| % of sales in Industry 3 | 14% | 16% | 17% | 19% |
| % of sales in Industry 4 | 16% | 15% | 19% | 8% |
| % of sales in Other Industries | 5% | 10% | 10% | 6% |
| % of sales in Region I | 5% | 3% | 5% | 13% |
| % of sales in Region II | 8% | 4% | 71% | 14% |
| % of sales in Region III | 10% | 6% | 10% | 54% |
| % of sales in Region IV | 73% | 3% | 8% | 13% |
| % of sales in Region V | 4% | 85% | 5% | 6% |

## 4.2. Predictive Model

In this section we present the summary of performance comparisons of all the predictive models that have been built followed by a more detailed discussion on the best performing model and its integration with the mathematical optimization model.

### 4.2.1. Models Evaluation and Selection

As mentioned in Methodology, we used mean square error (MSE) on both the train set and test set as the primary method for evaluating the performance of the four predictive models. The summary of the MSE result is displayed in Table 4-2. Neural Networks and XG-Boost models yield visibly superior performance than the rest of the algorithms.

*Table 4-2 Mean Squared Error of Predictive Models*

| Model | Trainset MSE | Testset MSE |
|---|---|---|
| Feed-forward Neural Network | 0.56 | 0.45 |
| XG-Boost | 0.49 | 0.50 |
| Revised Croston's | 1.22 | 1.15 |
| Random Forest | 1.97 | 1.42 |
| Linear Regression | 2.14 | 1.55 |

During our evaluation of the models, we also considered other factors such as ease of implementation, interpretability, and computation time. Although computation time increases for more complex algorithms, like the FNN model, we believe the gap in the performance of the model outweighs all other factors. In Figure 4-2, we analyzed the results in more detail, giving extra attention to the two best-performing algorithms to determine the most suitable model for the company. The FNN model excels at predicting spikes in demand, while the other model's predictions are much smoother. This phenomenon, along with its overall MSE score, makes the FNN the best model for the Company. In Introduction, we discuss about the lost sales related to the lack of equipment available at each branch. To prevent lost sales, the Company will be better suited to use the FNN model, because the model will better account for demand irregularities. In the next section, we will provide detailed results of the FNN model.

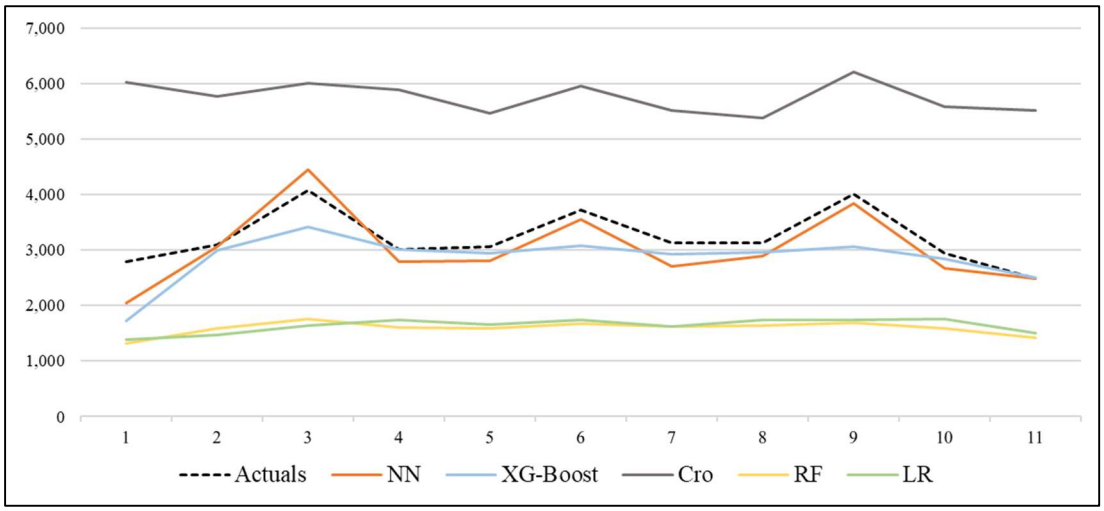


*Figure 4-2 Predictive Model Comparison*

### 4.2.2. Feed-forward Neural Network (FNN)

FNN hyperparameters tuning was performed to find the combination of number of hidden neurons, λ, and η which yields the lowest MSE score. Given the search space of 3×3×3=27 combinations, 5-fold cross validation, and maximum iterations of 50 epochs over the training set, the hyperparameters tuning and parameters training together ended up taking a computation time of about 48.8 hour. The best model returns a mean validation MSE of 0.56 with 10 neurons in the hidden layer, η=1e-3, and λ=1e-4. Figure 4-3 displays the mean validation MSE over the different combinations of tested parameters with the red marker signifying the best result. A more detailed set of records per combination can be seen in Appendix B.

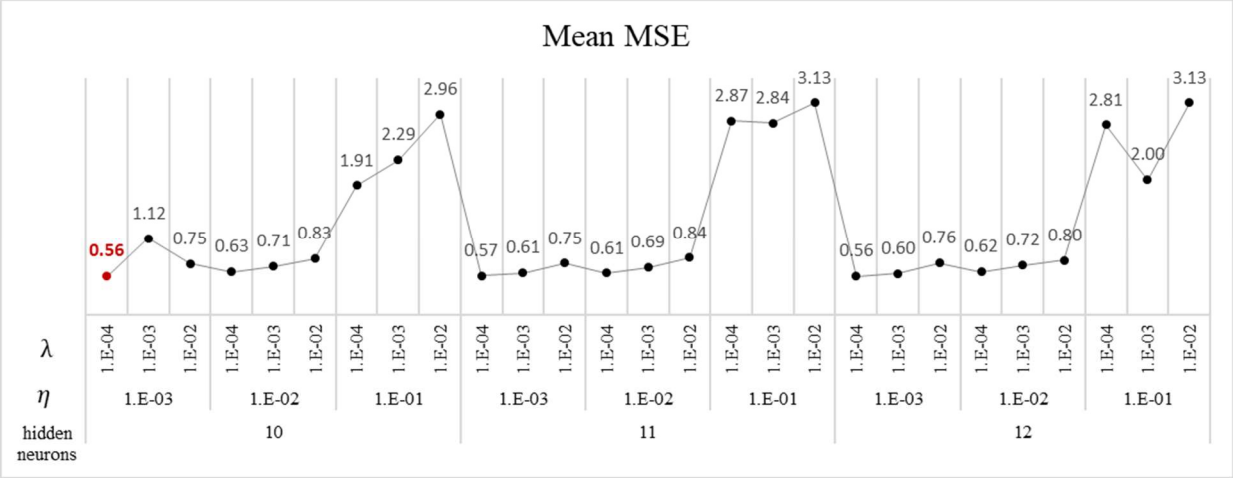


*Figure 4-3 Mean Validation MSE of Different Hyperparameters Combination*

When the best model is evaluated using the test dataset, which will be discussed later in this section, it returns an even lower MSE of 0.45. This low test MSE indicates an effective use of regularization term during training. Given the sufficiently low MSE value and the computationally expensive process of hyperparameter tuning, model training can be run in a yearly basis to keep the model up to date with a more strategic long-term change that might happen in demand behavior.

We use the connection weight approach as explained in the methodology section to interpret how each input feature contributes to the output of the FNN model. The connection weight is the summed product of input-to-hidden and hidden-to-output weights for each input feature. For brevity, Figure 4-4 and Figure 4-5 display features with absolute connection weight values of at least one which together represent 45% of total input features (30 out of 66) and 81% of the total sum of absolute connection weight values. Detailed tables containing all the connection weights, weights, and biases of the FNN model can be seen in Appendix C and D. For clarity of discussion, each of the feature mentioned in the following paragraphs will be written in a different `font style`.

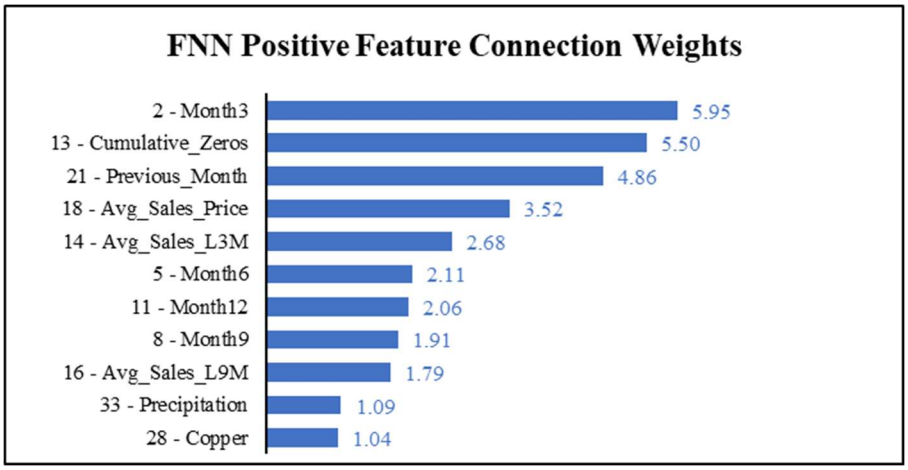


*Figure 4-5 FNN Positive Feature Connection Weights*

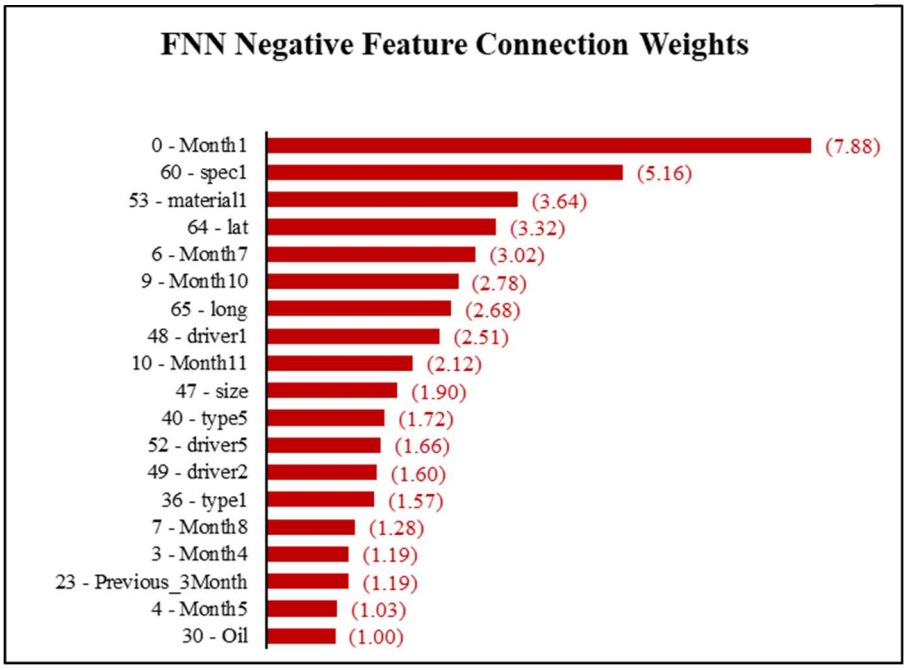


*Figure 4-4 FNN Negative Feature Connection Weights*

It is important to note that weights and connection weights in an FNN, as with many other forecast or predictive models, should not be interpreted as magnitude of causal relationship between feature input and output. They are, however, can be used to interpret how predicted demand is extrapolated from the interaction of feature inputs in the model. Among the prominent features represented in the FNN connection weights charts above, only three of them are exogenous features, namely `Precipitation` and `Copper` which correlate positively with output, and `Oil` which correlates negatively. Given the nature of the products involved in the business segment, which are water pumps, it is reasonable to see that more precipitation in a given period and place is mapped to more product demand, probably for flood-related emergency

or cautionary purposes. Futures price index for `Copper`, a mineral which is mostly used in building construction (U.S. Geological Survey, 2020), is also reasonably mapped to more demand in line with our discussions and site visits with the company's stakeholders where construction sector was understood to be one of the most recurrent customers of their business.

The rest of the features with prominent connection weights are either time-series or endogenous features and it is why we utilize only these features in the operational-predictive model to be delivered to the company. Furthermore, because of the practical convenience and reliability of interpolating time-series features as opposed to exogenous feature forecast. Of those with positive connection weights, it is notable that months at every quarter-end (`Month3`, `Month6`, `Month9`, and `Month12`) are present which is consistent with the seasonality spikes that we observed during initial data exploration. Elements of demand trend are translated into the model by `Average_Sales_L3M`, `Average_Sales_L9M`, and demand quantity on `Previous_Month`, while `Cumulative_Zeros` implies the intermittent nature of the demand. Lastly, we find it reasonable to see higher `Avg_Sales_Price` occurring concurrently with higher demand. This correlation happens possibly due to the classic supply-demand interaction with price and/or more involvement of equipment which are newer or has more features (therefore have higher prices) to fulfil customer demand as the cheaper alternatives has been rented out. However, interpreting it in the opposite direction(i.e., increasing `Avg_Sales_Price` can be one of the policy levers to drive demand up) is obviously misleading and therefore this feature should not be used for predictive purposes.

Among features with negative connection weights, we also see time-series elements which are represented, in order of magnitude, by `Month1`, `Month7`, `Month10`, `Month11`, `Month8`, `Month4`, `Previous_3_Month` and `Month5`. This indicates that the model maps lower demand to those particular months. The features `lat` and `long` respectively represent the latitude and longitude of the company's branches. Negative weight for latitude and longitude features of the branch means the model maps more demand quantity to branches on the more southerly and westerly geographical coordinates, as confirmed by the observation in Figure 3-7 during data preparation. The rest of the features with negative connection weight are endogenous specification of the product.

The true performance of the model is evaluated using the test dataset which is excluded from the dataset used for training the FNN model. A multi-perspective analysis of the model's

prediction versus actual values, therefore, helps us to dissect how reliable the model is in extrapolating demand quantity based on a set of yet unseen data features. In order to emulate the real-life implementation of this model, we round up each prediction value to its nearest integer. As expected, the resulting prediction values yield slightly higher overall MSE of 0.47 and RMSE of 0.68 compared to MSE of 0.45 from raw unrounded prediction values. We start looking at the overall prediction result by month as displayed in Figure 4-6 and by branch in figure 4-7. From monthly demand perspective, test set seems to largely conform with the quarterly seasonality of the preceding years' data in the training set. That is why the model is able to predict the general monthly fluctuation pattern although it consistently underestimates the demand.

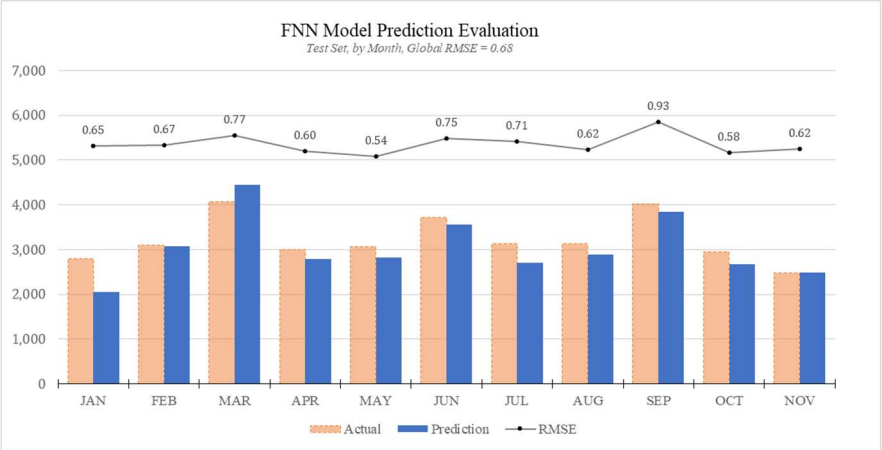


*Figure 4-6 FNN Model Prediction Evaluation by Month*

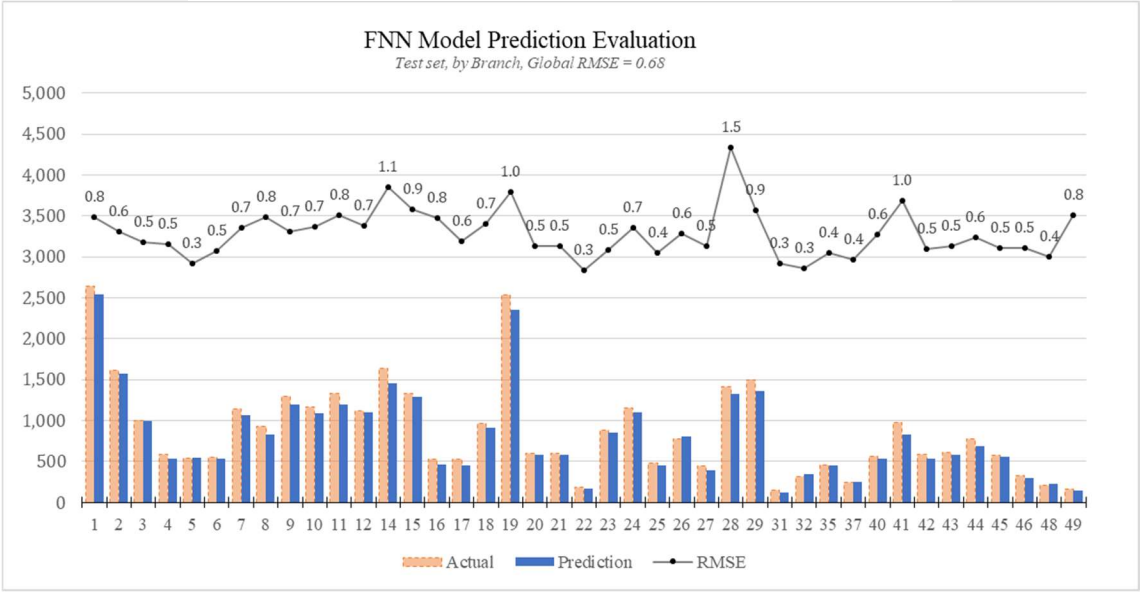


*Figure 4-7 FNN Model Prediction Evaluation by Branch*

Notable exceptions occur in March where the prediction exceeds actual value and in January where the gap between the prediction and actual value is the most prominent. From this observation we can infer that there are uncaptured factors and circumstances at, or leading up, to each of those two months in the test set period which caused demand to behave in a more pronounced anomaly relative to its historical behaviors. Such errors can always be expected from any predictive exercises and should function as a guidance in adjusting decisions made based on prediction data.

We also observe a tendency of error (RMSE) values to increase with higher demand values, as can be seen in the seasonal peak months of March, June, and September. The same behavior is also displayed in the prediction evaluation by branch, where error values largely follow the corresponding branch's demand level. Upon further investigation, we indeed found that for higher actual demand values there is a clear trend of increasing prediction error as can be seen in Figure 4-8. However, if we look at the distribution of actual demand values, 95% of them range from 0-2 where errors are still less than 1 and close to the global RMSE. Even if we exclude zero values from actual demand, a still very high proportion (80%) of the prediction yield error of less than 1. This increasing prediction error over larger demand quantity is expected due to the distribution of the training dataset where values of range 0-2 dominate the actual demand values (see Figure 3-3), giving the algorithm significantly more learning iterations around that range compared to larger values.

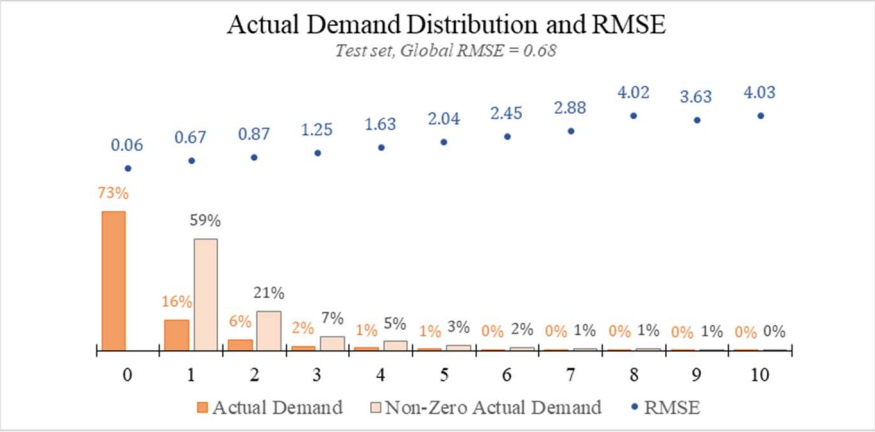


*Figure 4-8 Actual Demand Distribution and RMSE*

## 4.3. Optimization Model

As depicted in Figure 3-14, the mathematical model to optimize pump allocation takes in the resulting demand prediction from the FNN model as well as stock, lane capacity, and variable transportation cost as a set of input of a linear program and prescribes a set of transportation decisions as the final output for each month. If we represent the model, just like Gurobi does behind the scene, as a matrix, then it has 1,529,868 columns and 70,078 rows. The matrix columns represent all the decision variables $x_{kij}$ of the model, where we have a total of 723 items $k$, origin branch $i = 0$ has 46 possible lanes to all destination branches $j$, and each origin branch $i \neq 0$ has 45 possible lanes to all destination branches $j \neq i$ so $n(x_{kij}) = 723 \times (46 + (46 \times 45))$. The rows, on the other hand, represent all the constraints which are explicitly specified in the program, i.e., demand constraint (21) and supply constraint (22) respectively for $47 \times 723 = 33{,}981$ possible combination of branches and items, as well as lane capacity constraint (23) for $46 + (46 \times 45) = 2116$ lanes.

In order to evaluate the optimization model, we took the prediction result from the test dataset particularly the most recent month. We neither had access to a historical data at any period nor were we able to construct reliable assumption from the sporadic information in the available dataset related to stock position. For the sake of testing the logic of the model, we decided to generate random number using Excel for each predicted item demand $\hat{\delta}_{kj}$ to serve as hypothetical item stock position $\xi_{kj}$ at the end of the preceding month. The model took less than 3 minutes processing times before yielding results as depicted relationally in Figure 4-9. The main result table *solution* prescribes all the necessary item movement among origin and destination branches with their respective quantity, and variable cost. Table *net_demand* stores the stock and predicted demand input data which are used in the calculation of net demand and net supply as formulated in equation (26) and (27) respectively.

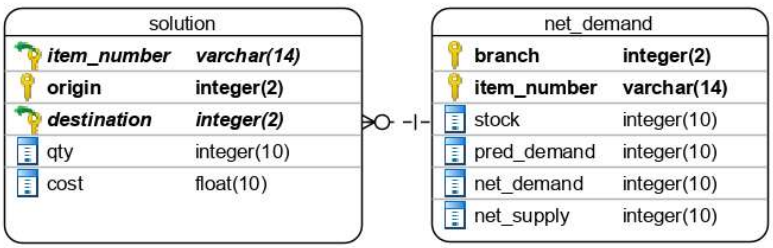


*Figure 4-9 Diagram of Optimization Model Outputs*

Due to the granularity of data in *solution* table and the same logic applied to all items and lanes, it is already insightful to evaluate how the model optimizes allocation in the item level based on several isolated samples. Let us take a look at the case of item number 1. In Table 4-3 we can see that as branch 9, 45, and 24 hold the stock of item 1 at the end of the preceding period, they have no predicted demand for the current period so each of them has net supply capability at the level of their respective stock. On the other hand, branch 19 holds only 1 item but has predicted demand of 11, so it needs another 10 as net demand in order to anticipate the realization of the prediction. In Table 4-3 *net_demand*, dummy branch 0 records the total stock and predicted demand of each item and only yields net supply capabilities whenever total predicted demand exceeds total stock as per equation (20).

*Table 4-3 Net Demand and Supply for Item Number 1*

| branch | item_number | stock | pred_demand | net_demand | net_supply |
|---|---|---|---|---|---|
| 9 | 1 | 9 | 0 | 0 | 9 |
| 45 | 1 | 8 | 0 | 0 | 8 |
| 24 | 1 | 4 | 0 | 0 | 4 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 19 | 1 | 1 | 11 | 10 | 0 |
| 0 | 1 | 22 | 11 | 0 | 0 |

With the information from Table 4-3, the model then needs to answer where the 10 units of net demand for branch 19 should come from among the branches with net supply i.e., 9, 45, or 24. From Table 4-4, we can see that branch 45 is the one with the lowest variable cost of 169.74 followed by 24 and 9.

*Table 4-4 Transportation Cost Between Branch 45, 24, 9 and 19*

| origin | destination | cost |
|---|---|---|
| 45 | 19 | 169.736015 |
| 24 | 19 | 495.2624156 |
| 9 | 19 | 697.0576059 |

With the net supply and demand capability specified in Table 4-3 and Transportation Cost in Table 4-4, the model returns a set of allocation decisions for item 1 as displayed in Table 4-5. We can see that the model maximizes the allocation from the branch with the lowest transportation cost to 19, which is 45, at 8 units which is the whole net supply capability of the

origin-branch. Thereafter, since the net demand for branch 19 is 10, the remaining 2 units are then prescribed to be sourced from the next lowest-cost branch 24. The column cost in table 4-5 denotes the total variable cost for the prescribed transportation, for example between branch 24 and 19, the cost is $495.26 \times 2 = 990.52$.

*Table 4-5 Prescribed Allocation Decision for Item Number 1*

| item_number | origin | destination | qty | cost |
|---|---|---|---|---|
| 1 | 24 | 19 | 2 | 990.52 |
| 1 | 45 | 19 | 8 | 1357.92 |

Another sample from the simulated result shows us how the model deals with the possibility of aggregate stock out, in other words total recorded stock of an item across the US is lower than total predicted demand. We take an item number 2, with the net demand situation as displayed in Table 4-6. There are only 4 stock of this item in branch 28, and demand is predicted to be 5. The model then assigns the net supply capability of the extra needed item to dummy branch 0 to satisfy the mathematical constraint of the model. Dummy branch 0 can only have net supply when total demand exceeds total recorded stock for a particular item. As expected, the prescribed allocation shown in Table 4-7 specifies a hypothetical transport from branch 0 to 28 to fulfil the demand. In practical implementation, the company can refer to items assigned to this dummy branch as a guidance to determine which items need replenishment or addition to the operational stock due to tendency of demand growth. From the simulation, only about 2% of the total pump movement need to be allocated from this dummy branch.

*Table 4-6 Net Demand and Supply for Item Number 2*

| branch | item_number | stock | pred_demand | net_demand | net_supply |
|---|---|---|---|---|---|
| 28 | 2 | 4 | 5 | 1 | 0 |
| 0 | 2 | 4 | 5 | 0 | 1 |

*Table 4-7 Prescribed Allocation Decision for Item Number 2*

| item_number | origin | destination | qty | cost |
|---|---|---|---|---|
| 2 | 0 | 28 | 1 | 9999 |

Based on the observation of these item samples, we conclude that the optimization model successfully fulfils its role to prescribe efficient allocation decision with reliable decision logic.

## 5. CONCLUSION

At the start of our research project, we learned about the company through discussions with senior management and site visits to several branch locations. We formulated our problem statement with the help of the stakeholders from the sponsoring company. The goal of our project was to increase asset utilization while at the same time, efficiently mobilizing their equipment. This led us to research the industrial equipment rental and water infrastructure industries as well as machine learning and mathematical optimization techniques. We proposed a methodology comparing four machine learning algorithms for forecasting demand, selecting the best algorithm, and feeding the predicted demand into a mathematical optimization model as an integrated decision tool for the company to manage its asset allocation. During the initial data preparation phase, we observed intermittent demand nature of the products and based on that observation we created several time-series features to be trained in the predictive models. After building and testing each predictive model, we chose the FNN model based on its overall predictive performance. The final step of our project was to incorporate the predictions of the FNN model to the optimization model to produce asset movement prescription.

Before the implementation of this project there has been no regular use of a sophisticated forecasting or optimization model at the Company. Most of the decisions made were reactionary and based on intuition. The FNN model that we developed objectively provides the company with a more data-driven prediction of their demand. On top of that, the optimization model utilizes both the demand prediction from FNN and stock position update from the company's current system to allocate pump assets in the most efficient way in terms of transportation miles. The simultaneous implementation of both models will bring the company in the direction where asset utilization increases and mobilization cost decreases. An integrated decision tool of both predictive and prescriptive optimization model is the main delivery of this project and the technical user guide is available in Appendix A for the company's reference.

Our models were developed within a certain boundary based on what we had access to during the research. The forecast models rely on historical sales and do not capture lost sales. The Company should consider lost sales from out-of-stocks situation into demand prediction model whenever such data are available in a consistent basis. Based on data availability and agreement with company leaders, demand prediction is aggregated in monthly basis in our models. In reality, there might be service areas and outlier circumstances where inventory need

to be reallocated weekly in the network. Selecting the appropriate aggregation level is an assessment of the tradeoff in forecast error and agility. Shifting to a weekly forecast will allow for rapid changes in asset allocations but diminish the forecast accuracy. Customer base and weather have influence on the sporadic nature of the business. Forecasting for rainfall, natural disasters, and local government spending is challenging. We included these factors in trying to infer strong predictive potential but based on the modeling result and practical reasons we do not include these exogenous features in the final prediction model in the integrated tool. In this project, we focus on the main pump product, but other product items will also benefit more from better demand planning and should be considered for future extension of this research project.

During conversations with leaders at the company, the stable growth prospects and high margin nature of the industry has caused the consolidation of several regional competitors. The attractive market also attracts the entry of strong national players with strength in technological innovation. These nuances in the competitive landscape are not captured in the model and can potentially affect future demand. As evidenced by the feature importance analysis in Section 4 Result and Discussion, there is also a strong indication for relationship between sales price and demand which due to the scope of this project was not explored deeply. These two areas, competition and pricing, might as well serve as interesting areas of future research to mine additional insight on how the industry's market dynamics affect demand behavior and how it can be defined in the predictive model.

## REFERENCES


Asala, H. I., Chebeir, J., Zhu, W., Gupta, I., Taleghani, A. D., & Romagnoli, J. (2017). A Machine Learning Approach to Optimize Shale Gas Supply Chain Networks. *SPE Annual Technical Conference and Exhibition*. SPE Annual Technical Conference and Exhibition, San Antonio, Texas, USA. https://doi.org/10.2118/187361-MS

Bluefield Research. (2019, April). *How Digital is Transforming the Future of Water* [Analyst Presentation]. The Design-Build for Water/Wastewater Conference, Cincinnati, OH.

Carbonneau, R., Laframboise, K., & Vahidov, R. (2008). Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, *184*(3), 1140–1154. https://doi.org/10.1016/j.ejor.2006.12.004

Croston, J. D. (1972). Forecasting and Stock Control for Intermittent Demands. *Operational Research Quarterly*, *23*(3), 16. http://dx.doi.org/10.1057/jors.1972.50.

Federal Reserve Bank of St. Louis. (2019). *Federal Reserve Economic Data* [Government Website]. Federal Reserve Economic Data. https://fred.stlouisfed.org/

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems* (Barker Library - Stacks Q325.5.G46 2019). Sebastopol, CA : O'Reilly Media, Inc., 2019.

Hardesty, L. (2017, April 14). *Explained: Neural networks*. MIT News. http://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

Islam, S., Keung, J., Lee, K., & Liu, A. (2012). Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, *28*(1), 155–162. https://doi.org/10.1016/j.future.2011.05.027

Jaakkola, T., & Barzilay, R. (2016). *Introduction to Machine Learning: Draft Notes by Topic* [Lecture Notes].

Jacquillat, A. (2019). *Analytics Edge* [Lecture Notes].

Kuraria, A., Jharbade, N., & Soni, M. (2018). Centroid Selection Process Using WCSS and Elbow Method for K-Mean Clustering Algorithm in Data Mining. *International Journal of Scientific Research in Science, Engineering and Technology*, 190–195. https://doi.org/10.32628/IJSRSET21841122

Lee, I., & Shin, Y. J. (2020). Machine learning for enterprises: Applications, algorithm selection, and challenges. *Business Horizons*, *63*(2), 157–170. https://doi.org/10.1016/j.bushor.2019.10.005

Lolli, F., Gamberini, R., Regattieri, A., Balugani, E., Gatos, T., & Gucci, S. (2017). Single-hidden layer neural networks for forecasting intermittent demand. *International Journal of Production Economics*, *183*, 116–128. https://doi.org/10.1016/j.ijpe.2016.10.021

López Lázaro, J., Barbero Jiménez, Á., & Takeda, A. (2018). Improving cash logistics in bank branches by coupling machine learning and robust optimization. *Expert Systems with Applications*, *92*, 236–255. https://doi.org/10.1016/j.eswa.2017.09.043

Maucec, M., & Garni, S. (2019). Application of Automated Machine Learning for Multi-Variate Prediction of Well Production. *SPE Middle East Oil and Gas Show and Conference*. SPE Middle East Oil and Gas Show and Conference, Manama, Bahrain. https://doi.org/10.2118/195022-MS

National Centers for Environmental Information (NCEI). (2019). *Climate at a Glance* [Government Website]. National Oceanic and Atmospheric Administration (NOAA). https://www.ncdc.noaa.gov/cag/statewide/time-series

National Oceanic and Atmospheric Administration. (2020). *Tropical Cyclone Climatology*. National Hurricane Center and Central Pacific Hurricane Center. https://www.nhc.noaa.gov/climo/

Nawar, S., & Mouazen, A. M. (2017). Comparison between Random Forests, Artificial Neural Networks and Gradient Boosted Machines Methods of On-Line Vis-NIR Spectroscopy Measurements of Soil Total Nitrogen and Total Carbon. *Sensors (Basel, Switzerland)*, *17*(10). https://doi.org/10.3390/s17102428

Olden, J. D., & Jackson, D. A. (2002). Illuminating the “black box”: A randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling*, *154*(1–2), 135–150. https://doi.org/10.1016/S0304-3800(02)00064-9

Olden, J. D., Joy, M. K., & Death, R. G. (2004). An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. *Ecological Modelling*, *178*(3–4), 389–397. https://doi.org/10.1016/j.ecolmodel.2004.03.013

Roth, R. (2019). *Industrial Equipment Rental & Leasing in the US* (Industry Report No. 53249; IBISWorld Industry Report). IBISWorld. https://my.ibisworld.com/us/en/industry/53249/about

Silver, E. A., Pyke, D. F., & Thomas, D. J. (2016a). Cost and Other Important Factors. In *Inventory and Production Management in Supply Chains* (pp. 40–44). CRC Press. https://doi.org/10.1201/9781315374406

Silver, E. A., Pyke, D. F., & Thomas, D. J. (2016b). Intermittent and Erratic Demand. In *Inventory and Production Management in Supply Chains* (pp. 122–123). CRC Press. https://doi.org/10.1201/9781315374406

Syntetos, A. A., & Boylan, J. E. (2001). On the bias of intermittent demand estimates. *International Journal of Production Economics*, *71*(1–3), 457–466. https://doi.org/10.1016/S0925-5273(00)00143-2

U.S. Geological Survey. (2020). *Mineral commodity summaries 2020: U.S. Geological Survey,* (p. 200). U.S. Geological Survey. https://doi.org/10.3133/mcs2020

Ycharts. (2019). *Commodities Futures Prices*. YCharts. https://get.ycharts.com/solutions/

# APPENDIX A – DECISION TOOL USER GUIDE

## I. Software and System Requirement

The decision tool was written in Python 3.7 programming language and structurally designed to be run most comfortably using Jupyter Notebook ("Notebook") web application. The Python codes run in a computer with either Mac or Windows operating system, has Microsoft Excel or other application which is able to read a .csv file, and an internet browser.

The first software that needs to be installed is Anaconda, we used Anaconda Individual for the development of the decision tool but you can also use Team or Enterprise edition if you wish. Download the appropriate Anaconda installer with Python 3.7 version for your OS from this page https://www.anaconda.com/distribution/. Visit https://docs.anaconda.com/anaconda/ for installation guide, user guide, and other references.

To preprocess the data from the company's system, Alteryx, an extract-transform-load (ETL) software tool was also used. The user can download and subscribe to the software at https://www.alteryx.com/platform.

For optimization, we used Gurobi optimizer software in Python interface. Gurobi optimizer can be downloaded from this site https://www.gurobi.com/downloads/gurobi-optimizer-eula/. For commercial use, the company will need to invest in license fee for the product.

Once Anaconda is installed in your computer, search for Anaconda Navigator, open it, and go to tab Environments > base(root) and in the right most section of the window click the scroll down

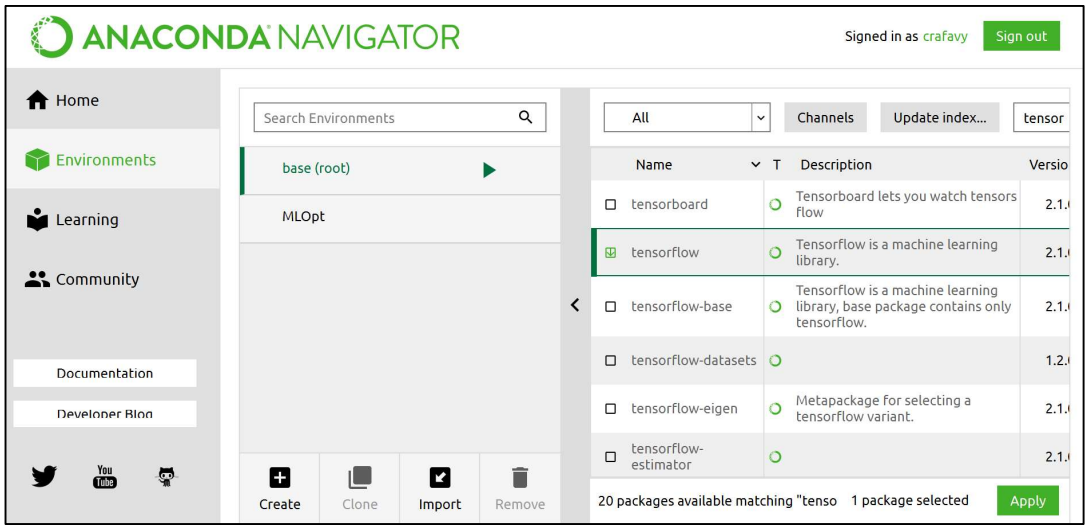


*Figure A-1 Anaconda Navigator Environments Tab Screenshot*

bar and choose All. On the search box, type to find these libraries: *tensorflow, keras, scikit-learn*, and *gurobi* in the list and click the check box for each library till it displays a downward arrow. After all three libraries are checked, click Apply to download and install them in your Anaconda environment. Although it should be a default setting, you might also want to check if *pandas* and *numpy* libraries have already had checked box in the list, in which case they are already callable for use in Python codes.

Once installations are finished, go to Home tab find Jupyter Notebook in the scroll down area. Click the Install button for the Notebook or Launch button when it has already been installed. Jupyter Notebook is a web application so it will open and run in a web browser so manually open your web browser if it does not automatically open upon clicking the launch button for the notebook in Anaconda Navigator.

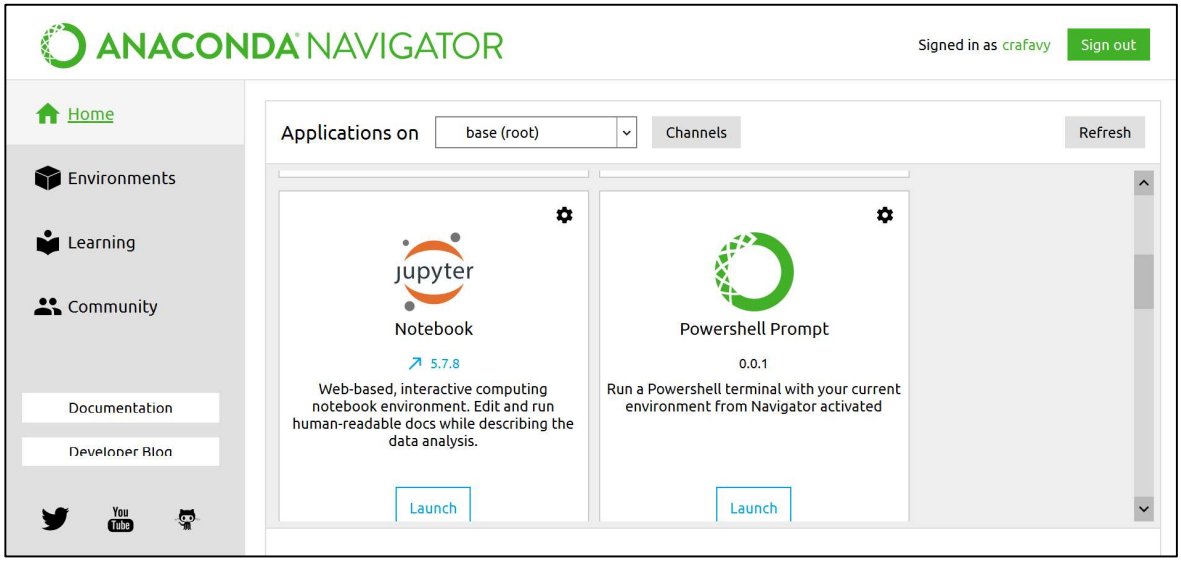


*Figure A-2 Anaconda Navigator Home Tab Screenshot*

When Jupyter Notebook opens in the web browser, you will see the navigation page which contains the directory of your computer drives. Go to the directory where you store the decision tool package folder named MODEL.

## II. File Structure

All the necessary input and output files in .csv as well as the decision tool's code skeleton itself are contained and structured in a folder we name MODEL. You should not relocate, rename, or delete any of the pre-existing folders and files inside if you want the tool to work properly.

Although you can add new files anywhere inside the MODEL folder, we highly recommend that you store any of them outside the folder to maintain data structure integrity of the tool.

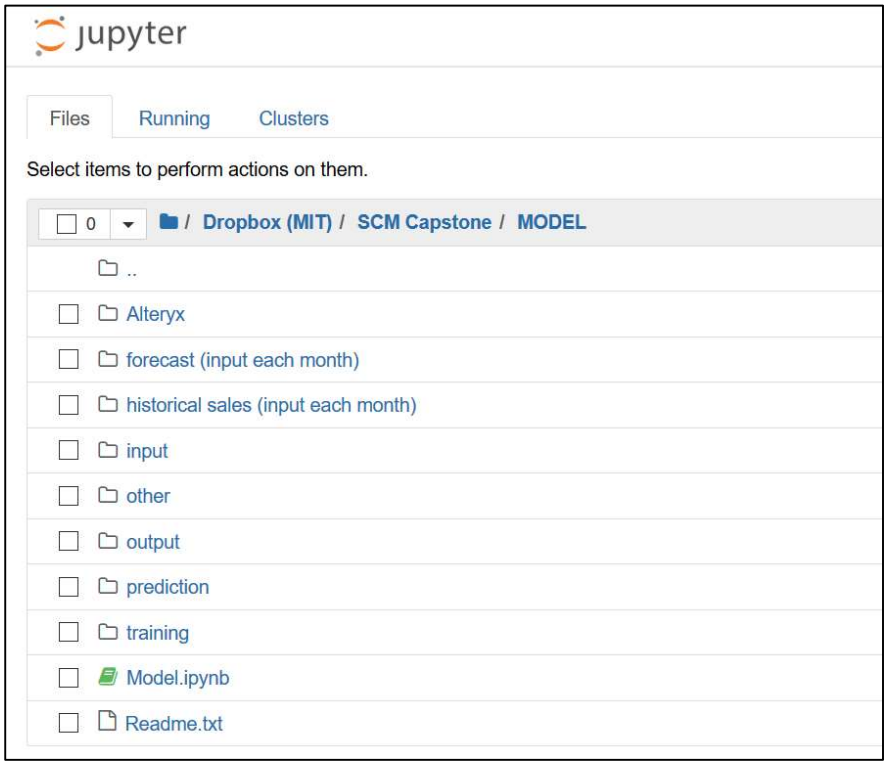


*Figure A-3 Jupyter Notebook Navigation Page Screenshot*

The MODEL folder structure is as follows:

**MODEL**

*Model.ipynb*

*Readme.txt*

**input:** *branch.csv, lane.csv, stock.csv*

**output:** *net_demand.csv, solution.csv*

**prediction:** *prediction.csv*

**training:** *bias_hidden.csv, bias_output.csv, dataset.csv, evaluation.csv, fnn.h5, grid_cv_results.csv, weight_hidden.csv, weight_input.csv, scope.csv*

**Alteryx:** *Wflow_1.yxmd, Wflow_2.yxmd, Wflow_Consolidated.yxdb, Wflow_2.bak*

**forecast (input each month)**: *forecast input.xlsx*

**historical sales (input each month):** contains historical sales data

*Model.ipynb* is the python notebook object containing the codes for running the decision tool. In subfolder 'input', *branch.csv* contains the latitude and longitude of each branch, *lane.csv* contains all possible lanes between branches along with the latitude and longitude of each origin and destination branch as well as variable transportation cost and capacity of each lane, and *stock.csv* contains the stock position of each item number at each branch at the end of the most recent month. The subfolder 'output' contains the main output of the whole decision tool which is *solution.csv* where item movement quantity, origin, destination, and cost are prescribed.

Subfolder 'prediction' on the other hand contains the output of the predictive model *demand.csv* where the predicted demand of each item for each branch is stored. As depicted by the tool's working diagram in Figure 3-9 in the main body of the project's report, you can manually adjust the predicted demand values in *demand.csv* to incorporate business judgment as necessary before the data are fed into the optimization model. The subfolder *training* contains the main feed-forward neural networks (FNN) model *fnn.h5*, the historical data *dataset.csv* used for both forecasting and FNN training purpose, the structural parameters of current FNN (*bias_hidden.csv, bias_output.csv, weight_hidden.csv, weight_input.csv*), the summary result of the latest FNN training session in *grid_cv_results.csv*, and *scope,csv* which contains the exhaustive list of Item Numbers and Branches which are included in the model.

Subfolders 'Alteryx', 'forecast (input each month)', and 'historical sales (input each month)' contain all the files and workflow template needed for pre-processing data retrieved from the company's system into *dataset.csv*. Pre-processing is needed to summarize sales quantity at item and branch level as well as generating all the time series features.

III. Data Pre-processing

At the end of every month, user should extract the most recent month's actual sales history (e.g., November of the test set period if we refer to the data involved in this project) from the company's database system and save it in subfolder 'historical sales (input each month)'.

After the most recent month's actual sales data is saved to the folder, open the Alteryx workflow titled *Wflow_1*. Upload the new actual sales file using the Input Data tool and join the month to the existing Union tool. Afterwards, run the workflow. The output will be titled *Wflow_Consolidated*.

Next, open the *forecast input* file in subfolder 'forecast (input each month)'. This file will also need to be updated every month to create new data rows for the forecast month. By default the file already contains all pairs of branch and item as defined in the scope of predictive models. User needs to update the 'Date', 'Month', and 'Year' field in the file according to the target forecast period. The *Date* field should be written in the format "mm_yyyy" as shown in the example in the file. Save the file.

## IV. Implementation

Python implementation is performed only after Data Pre-processing is done. Open *Model.ipynb* from the Jupyter Notebook's Navigation Page to access the skeleton code of the decision tool . It should look like the screenshot in Figure A-7.

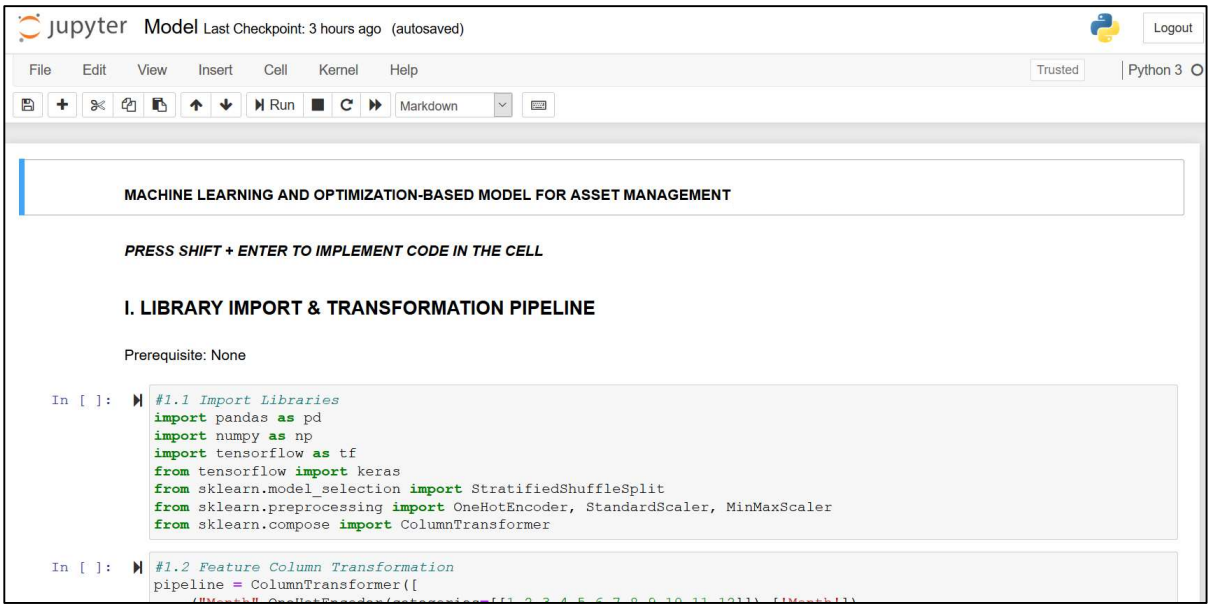


*Figure A-7 Decision Tool's Code Skeleton in Jupyter Notebook Screenshot*

A Jupyter Notebook has greyed out sections where codes are written which are called cells and as explicitly hinted in the Notebook, you should put your cursor inside a cell and press shift and center to run the code lines in a cell. The message `In [ ]` on the left side of each cell indicates the sequence order of that cell's implementation. For example, `In [2]` means that the cell has been implemented right after the cell with `In [1]` besides it. There are 4 sections of code execution in the Notebook and in each sections, code lines are partitioned by numbered comment

lines which looks like this: `#1.1 Import Libraries` to help you and future developers understand the steps that are taken when the codes are run. The 4 sections are as follows:

I. Library Import and Transformation Pipeline

   This section is the prerequisite for all the other sections. Make sure all cells under this section have been implemented before running any other section.

II. Monthly Forecasting

   Run this section only if section I has been implemented. Specify the month and year of the forecast period in part `2.1 Specify Forecasting Period` and then run the rest of the codes in the section. This section outputs *demand.csv* in folder prediction, which you can inspect and adjust manually whenever necessary before running the next code section. This section should be run in a monthly basis at the end of the most recent month before the forecast month. Make sure that all the necessary updates on the data stored in folder 'input' have been deployed before forecasting.

III. Allocation Optimization

   Run this section only if section I and section II have been implemented. This section produces the final output of the whole model *solution.csv* which is accessible from the subfolder 'output'.

IV. FNN Training

   Run this section only if section I has been implemented. This section is the most computationally expensive of the whole model since it involves training the whole dataset through multiple epochs, cross validation, and combination of FNN hyperparameters (number of hidden neurons, learning rate, and l2 regularization) to ensure robust predictive capability. With the hardware specification we used during development, it took almost 48 hours in total to run the training so prepare in advance and make sure the machine is connected to power source the whole time. When the codes are run, you can still use other applications in the computer but you have to keep the browser displaying the Notebook open. Run the training at the beginning of each year and enter the preceding year as the `test year` in part `#4.1 Specify Dataset Year to be split out of the main dataset as Test Set`. The default search space for each hyperparameters is specified in part `#4.2 Specify search space for hyperparameters` but you have the option of narrowing it down

to fewer ranges to save computation time if you see that the operational model still returns satisfactory forecast accuracy. Afterwards, run the codes in rest of the section to get the new model *fnn.h5* which is stored in subfolder 'training' along with the FNN model parameter specifications as discussed in File Structure earlier.

## APPENDIX B – FNN HYPERPARAMETER TUNING CROSS-VALIDATION RESULT

| computation time (s) | hyperparameters combination | test_score[1] | | | | | | | train_score[1] | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | split0 | split1 | split2 | split3 | split4 | mean | rank | split0 | split1 | split2 | split3 | split4 | mean |
| 8,972 | **{'l2': 0.0001, 'lr': 0.001, 'neurons': 10}** | -0.53 | -0.52 | -0.54 | -0.60 | -0.62 | **-0.56** | **1** | -0.56 | -0.55 | -0.56 | -0.52 | -0.55 | -0.55 |
| 6,921 | {'l2': 0.0001, 'lr': 0.001, 'neurons': 12} | -0.54 | -0.53 | -0.53 | -0.62 | -0.61 | -0.56 | 2 | -0.54 | -0.56 | -0.54 | -0.54 | -0.53 | -0.54 |
| 7,869 | {'l2': 0.0001, 'lr': 0.001, 'neurons': 11} | -0.53 | -0.55 | -0.53 | -0.61 | -0.62 | -0.57 | 3 | -0.55 | -0.58 | -0.54 | -0.53 | -0.54 | -0.55 |
| 8,775 | {'l2': 0.001, 'lr': 0.001, 'neurons': 12} | -0.58 | -0.56 | -0.57 | -0.66 | -0.64 | -0.60 | 4 | -0.60 | -0.59 | -0.59 | -0.58 | -0.57 | -0.59 |
| 3,739 | {'l2': 0.0001, 'lr': 0.01, 'neurons': 11} | -0.58 | -0.54 | -0.60 | -0.65 | -0.65 | -0.61 | 5 | -0.62 | -0.59 | -0.63 | -0.58 | -0.56 | -0.59 |
| 7,100 | {'l2': 0.001, 'lr': 0.001, 'neurons': 11} | -0.59 | -0.60 | -0.57 | -0.65 | -0.64 | -0.61 | 6 | -0.62 | -0.64 | -0.60 | -0.57 | -0.57 | -0.60 |
| 2,488 | {'l2': 0.0001, 'lr': 0.01, 'neurons': 12} | -0.59 | -0.62 | -0.58 | -0.69 | -0.64 | -0.62 | 7 | -0.61 | -0.65 | -0.61 | -0.61 | -0.58 | -0.61 |
| 30,904 | {'l2': 0.0001, 'lr': 0.01, 'neurons': 10} | -0.60 | -0.59 | -0.62 | -0.67 | -0.65 | -0.63 | 8 | -0.65 | -0.62 | -0.61 | -0.59 | -0.58 | -0.61 |
| 2,898 | {'l2': 0.001, 'lr': 0.01, 'neurons': 11} | -0.64 | -0.63 | -0.66 | -0.83 | -0.70 | -0.69 | 9 | -0.69 | -0.67 | -0.70 | -0.73 | -0.63 | -0.68 |
| 3,071 | {'l2': 0.001, 'lr': 0.01, 'neurons': 10} | -0.70 | -0.65 | -0.71 | -0.76 | -0.71 | -0.71 | 10 | -0.73 | -0.69 | -0.74 | -0.67 | -0.65 | -0.70 |
| 3,293 | {'l2': 0.001, 'lr': 0.01, 'neurons': 12} | -0.62 | -0.64 | -0.63 | -1.02 | -0.72 | -0.72 | 11 | -0.70 | -0.69 | -0.67 | -0.93 | -0.64 | -0.72 |
| 10,025 | {'l2': 0.01, 'lr': 0.001, 'neurons': 10} | -0.69 | -0.72 | -0.72 | -0.82 | -0.81 | -0.75 | 12 | -0.77 | -0.75 | -0.75 | -0.72 | -0.74 | -0.74 |
| 8,287 | {'l2': 0.01, 'lr': 0.001, 'neurons': 11} | -0.69 | -0.71 | -0.72 | -0.82 | -0.82 | -0.75 | 13 | -0.77 | -0.75 | -0.75 | -0.72 | -0.75 | -0.75 |
| 10,849 | {'l2': 0.01, 'lr': 0.001, 'neurons': 12} | -0.69 | -0.72 | -0.73 | -0.84 | -0.81 | -0.76 | 14 | -0.77 | -0.76 | -0.76 | -0.74 | -0.73 | -0.75 |
| 3,168 | {'l2': 0.01, 'lr': 0.01, 'neurons': 12} | -0.72 | -0.77 | -0.81 | -0.85 | -0.83 | -0.80 | 15 | -0.82 | -0.81 | -0.84 | -0.75 | -0.74 | -0.79 |
| 2,268 | {'l2': 0.01, 'lr': 0.01, 'neurons': 10} | -0.83 | -0.74 | -0.80 | -0.91 | -0.85 | -0.83 | 16 | -0.92 | -0.78 | -0.84 | -0.81 | -0.76 | -0.82 |
| 3,011 | {'l2': 0.01, 'lr': 0.01, 'neurons': 11} | -0.71 | -0.85 | -0.78 | -0.98 | -0.85 | -0.84 | 17 | -0.81 | -0.90 | -0.81 | -0.87 | -0.76 | -0.83 |
| 8,147 | {'l2': 0.001, 'lr': 0.001, 'neurons': 10} | -0.59 | -3.13 | -0.58 | -0.68 | -0.64 | -1.12 | 18 | -0.64 | -3.14 | -0.59 | -0.59 | -0.57 | -1.11 |
| 5,061 | {'l2': 0.0001, 'lr': 0.1, 'neurons': 10} | -1.29 | -1.47 | -1.99 | -1.56 | -3.25 | -1.91 | 19 | -1.49 | -1.50 | -1.93 | -1.46 | -3.10 | -1.90 |
| 2,395 | {'l2': 0.001, 'lr': 0.1, 'neurons': 12} | -1.60 | -2.02 | -1.51 | -1.62 | -3.25 | -2.00 | 20 | -1.84 | -2.04 | -1.50 | -1.52 | -3.10 | -2.00 |
| 2,581 | {'l2': 0.001, 'lr': 0.1, 'neurons': 10} | -2.84 | -3.12 | -1.57 | -2.22 | -1.69 | -2.29 | 21 | -3.20 | -3.13 | -1.58 | -2.11 | -1.57 | -2.32 |
| 13,324 | {'l2': 0.0001, 'lr': 0.1, 'neurons': 12} | -2.84 | -3.12 | -3.25 | -1.56 | -3.25 | -2.81 | 22 | -3.20 | -3.13 | -3.10 | -1.45 | -3.10 | -2.80 |
| 2,797 | {'l2': 0.001, 'lr': 0.1, 'neurons': 11} | -2.84 | -3.12 | -3.25 | -1.72 | -3.25 | -2.84 | 23 | -3.20 | -3.13 | -3.10 | -1.62 | -3.10 | -2.83 |
| 12,055 | {'l2': 0.0001, 'lr': 0.1, 'neurons': 11} | -2.84 | -3.12 | -3.25 | -1.86 | -3.25 | -2.87 | 24 | -3.20 | -3.13 | -3.10 | -1.77 | -3.10 | -2.86 |
| 1,935 | {'l2': 0.01, 'lr': 0.1, 'neurons': 10} | -1.99 | -3.12 | -3.25 | -3.19 | -3.25 | -2.96 | 25 | -2.23 | -3.13 | -3.10 | -3.12 | -3.10 | -2.94 |
| 1,918 | {'l2': 0.01, 'lr': 0.1, 'neurons': 11} | -2.84 | -3.12 | -3.25 | -3.19 | -3.25 | -3.13 | 26 | -3.20 | -3.13 | -3.10 | -3.12 | -3.10 | -3.13 |
| 1,720 | {'l2': 0.01, 'lr': 0.1, 'neurons': 12} | -2.84 | -3.12 | -3.25 | -3.19 | -3.25 | -3.13 | 26 | -3.20 | -3.13 | -3.10 | -3.12 | -3.10 | -3.13 |
| **175,570** | | | | | | | | | | | | | | |

[1]rounded to 2 decimal places

# APPENDIX C – FNN WEIGHTS

| Index | Connection Weight[1] ($Ix_j$) | Hidden-Output Weight[1] ($V_k$) | -2.10 | 0.66 | -0.95 | 0.04 | -0.12 | 2.50 | 1.71 | 0.90 | 0.75 | -0.11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Input Feature ($x_j$) | Input-Hidden Weight[1] ($W_{j,k}$) | | | | | | | | | |
| | | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| 0 | -7.88 | Month1 | 0.22 | -0.67 | 0.25 | -0.18 | -0.19 | -1.53 | -1.36 | -0.85 | 0.24 | -0.10 |
| 1 | 0.53 | Month2 | 0.02 | 0.00 | 0.22 | -0.05 | 0.26 | 0.06 | 0.29 | 0.07 | 0.10 | -0.25 |
| 2 | 5.95 | Month3 | -0.05 | 0.10 | -0.05 | -0.14 | -0.16 | 1.27 | 1.07 | 0.61 | 0.16 | -0.25 |
| 3 | -1.19 | Month4 | -0.01 | 0.11 | 0.29 | 0.23 | 0.19 | -0.33 | -0.09 | -0.12 | 0.12 | -0.07 |
| 4 | -1.03 | Month5 | -0.04 | -0.07 | 0.10 | -0.12 | 0.14 | -0.25 | -0.22 | -0.02 | 0.09 | 0.08 |
| 5 | 2.11 | Month6 | -0.05 | 0.06 | -0.15 | -0.18 | -0.19 | 0.32 | 0.45 | 0.19 | 0.12 | 0.07 |
| 6 | -3.02 | Month7 | 0.04 | 0.11 | 0.16 | 0.00 | 0.10 | -0.64 | -0.55 | -0.19 | -0.16 | 0.10 |
| 7 | -1.28 | Month8 | -0.02 | -0.16 | 0.07 | 0.13 | -0.18 | -0.34 | -0.24 | -0.14 | 0.28 | -0.09 |
| 8 | 1.91 | Month9 | 0.03 | 0.07 | -0.03 | -0.13 | -0.26 | 0.24 | 0.48 | 0.32 | 0.20 | -0.15 |
| 9 | -2.78 | Month10 | -0.10 | -0.02 | 0.02 | 0.24 | 0.17 | -0.73 | -0.48 | -0.39 | 0.09 | 0.04 |
| 10 | -2.12 | Month11 | 0.04 | -0.12 | 0.21 | 0.21 | 0.18 | -0.55 | -0.26 | -0.16 | 0.30 | 0.00 |
| 11 | 2.06 | Month12 | 0.01 | 0.07 | 0.03 | -0.17 | 0.05 | 0.39 | 0.48 | 0.32 | -0.02 | 0.03 |
| 12 | -0.98 | Non_Zero_Demand_Intervals | 0.00 | 0.02 | 0.01 | 0.21 | -0.12 | -0.32 | -0.04 | 0.03 | -0.22 | 0.10 |
| 13 | 5.50 | Cumulative_Zeros | -0.01 | 0.20 | -0.06 | 0.15 | -0.08 | 1.28 | 1.00 | 0.52 | -0.11 | 0.02 |
| 14 | 2.68 | Avg_Sales_L3M | -0.05 | 0.11 | -0.17 | 0.01 | 0.09 | 0.41 | 0.40 | 0.69 | 0.06 | 0.16 |
| 15 | 0.82 | Avg_Sales_L6M | 0.03 | 0.16 | 0.00 | -0.06 | -0.12 | 0.25 | 0.09 | 0.09 | -0.16 | -0.23 |
| 16 | 1.79 | Avg_Sales_L9M | 0.02 | 0.16 | 0.09 | -0.21 | -0.06 | 0.47 | 0.28 | 0.10 | 0.12 | 0.08 |
| 17 | -0.64 | Average_Sales_LTM | -0.11 | 0.17 | -0.15 | -0.17 | -0.24 | -0.29 | -0.11 | -0.02 | -0.30 | -0.04 |
| 18 | 3.52 | Avg_Sales_Price | -2.71 | 1.40 | 1.83 | 0.22 | -0.20 | -0.88 | -0.02 | 0.07 | 0.97 | -0.36 |
| 19 | 0.50 | P1Y_Month | 0.03 | 0.22 | 0.05 | 0.03 | 0.12 | 0.21 | 0.02 | 0.02 | -0.10 | 0.06 |
| 20 | -0.01 | P2Y_Month | -0.01 | 0.13 | -0.02 | -0.13 | 0.03 | -0.02 | -0.01 | 0.00 | -0.11 | -0.09 |
| 21 | 4.86 | Previous_Month | 0.03 | -0.16 | 0.08 | -0.05 | -0.07 | 1.28 | 0.67 | 0.64 | 0.22 | -0.18 |
| 22 | -0.54 | Previous_2Month | 0.05 | -0.03 | 0.03 | -0.05 | 0.08 | -0.25 | 0.13 | 0.01 | 0.01 | -0.09 |
| 23 | -1.19 | Previous_3Month | 0.02 | 0.18 | 0.05 | 0.08 | -0.15 | -0.28 | -0.10 | -0.21 | -0.23 | 0.14 |
| 24 | -0.05 | YoY_Change_Avg_Sales_LTM | -0.02 | 0.00 | 0.00 | 0.20 | 0.17 | -0.03 | -0.02 | 0.01 | -0.01 | -0.16 |
| 25 | 0.46 | Construction_Spending_Sewage | -0.04 | 0.08 | 0.03 | 0.00 | -0.06 | 0.02 | 0.17 | 0.06 | -0.06 | 0.02 |
| 26 | 0.04 | Manf_New_Orders | 0.02 | -0.10 | -0.01 | -0.01 | -0.18 | 0.02 | -0.02 | -0.03 | 0.13 | -0.08 |
| 27 | -0.12 | Mining | 0.18 | 0.17 | 0.06 | 0.01 | 0.01 | 0.15 | -0.03 | 0.13 | -0.29 | -0.10 |
| 28 | 1.04 | Copper | -0.01 | 0.09 | 0.05 | -0.04 | 0.16 | 0.33 | 0.12 | 0.08 | -0.08 | 0.07 |
| 29 | 0.37 | Natural_Gas | 0.02 | -0.08 | 0.04 | -0.08 | 0.01 | 0.17 | 0.00 | -0.04 | 0.14 | -0.16 |
| 30 | -1.00 | Oil | -0.07 | -0.03 | -0.04 | 0.24 | 0.08 | -0.40 | -0.07 | -0.03 | -0.03 | 0.05 |
| 31 | 0.32 | Housing | -0.03 | -0.09 | 0.00 | -0.18 | -0.05 | 0.12 | 0.00 | 0.00 | 0.05 | 0.11 |
| 32 | 0.25 | ISM_PMI | 0.00 | -0.06 | -0.01 | 0.18 | -0.02 | 0.06 | 0.02 | 0.04 | 0.08 | 0.06 |
| 33 | 1.09 | Precipitation | 0.00 | 0.11 | 0.00 | -0.18 | -0.06 | 0.40 | 0.06 | 0.02 | -0.11 | 0.23 |
| 34 | -0.93 | Precip-Anomaly | 0.01 | 0.08 | 0.00 | 0.05 | 0.03 | -0.34 | -0.04 | -0.01 | -0.07 | -0.12 |
| 35 | 0.85 | Industrial_Manufacturing | -0.16 | 0.01 | -0.02 | -0.19 | 0.04 | 0.21 | 0.03 | -0.07 | -0.01 | 0.12 |
| 36 | -1.57 | type1 | 0.02 | 0.05 | 0.13 | 0.13 | -0.21 | -0.44 | -0.25 | -0.05 | 0.13 | -0.10 |
| 37 | -0.74 | type2 | 0.14 | 0.09 | 0.03 | -0.02 | 0.18 | 0.05 | -0.24 | -0.06 | -0.15 | 0.12 |
| 38 | -0.56 | type3 | -0.05 | -0.02 | 0.01 | -0.18 | -0.07 | -0.15 | -0.06 | -0.26 | 0.11 | 0.19 |
| 39 | -0.86 | type4 | -0.08 | 0.21 | 0.05 | 0.09 | 0.15 | -0.36 | -0.06 | -0.24 | 0.17 | 0.14 |
| 40 | -1.72 | type5 | 0.10 | -0.09 | -0.01 | -0.10 | 0.24 | -0.41 | -0.20 | -0.17 | 0.12 | -0.22 |
| 41 | -0.84 | type6 | 0.00 | 0.05 | 0.07 | -0.12 | -0.04 | -0.20 | -0.20 | 0.06 | -0.05 | -0.25 |
| 42 | -0.36 | type7 | 0.02 | -0.11 | -0.04 | 0.07 | 0.06 | -0.05 | -0.08 | 0.02 | 0.01 | 0.15 |
| 43 | -0.62 | type8 | 0.31 | 0.04 | -0.08 | 0.13 | 0.22 | 0.14 | -0.11 | -0.25 | 0.00 | -0.24 |
| 44 | -0.46 | type9 | -0.10 | -0.19 | -0.16 | -0.07 | -0.06 | -0.07 | -0.21 | -0.22 | -0.03 | -0.25 |
| 45 | 0.02 | type10 | -0.02 | -0.16 | -0.24 | -0.21 | 0.21 | -0.17 | 0.21 | 0.12 | -0.18 | 0.00 |
| 46 | -0.83 | type11 | -0.11 | 0.04 | -0.03 | 0.04 | 0.01 | -0.34 | -0.16 | -0.05 | 0.07 | -0.07 |
| 47 | -1.90 | size | 0.41 | -0.09 | -0.51 | 0.00 | 0.03 | -0.52 | -0.07 | -0.05 | 0.01 | -0.15 |
| 48 | -2.51 | driver1 | -0.08 | -0.13 | -0.08 | -0.19 | -0.08 | -0.79 | -0.36 | -0.18 | 0.06 | -0.21 |
| 49 | -1.60 | driver2 | -0.03 | -0.02 | 0.16 | -0.06 | -0.06 | -0.34 | -0.36 | -0.15 | 0.11 | -0.09 |
| 50 | -0.69 | driver3 | 0.06 | -0.16 | -0.07 | -0.18 | 0.12 | -0.11 | -0.05 | -0.20 | 0.05 | 0.06 |
| 51 | -0.77 | driver4 | 0.24 | 0.09 | 0.00 | -0.04 | 0.04 | -0.09 | -0.01 | -0.13 | 0.04 | -0.17 |
| 52 | -1.66 | driver5 | 0.17 | 0.20 | 0.17 | -0.17 | -0.09 | -0.39 | -0.01 | -0.22 | -0.06 | 0.17 |
| 53 | -3.64 | material1 | -0.06 | 0.06 | 0.07 | 0.03 | -0.25 | -1.15 | -0.56 | -0.02 | 0.07 | -0.12 |
| 54 | -0.52 | material2 | 0.06 | -0.05 | -0.03 | -0.26 | 0.23 | -0.10 | -0.05 | 0.06 | -0.07 | 0.23 |
| 55 | -0.25 | material3 | -0.14 | -0.11 | -0.09 | -0.12 | -0.25 | -0.20 | -0.04 | 0.18 | -0.24 | 0.00 |
| 56 | -0.83 | material4 | 0.20 | -0.01 | -0.03 | -0.26 | 0.01 | -0.20 | 0.08 | -0.21 | 0.21 | 0.18 |
| 57 | -0.02 | material5 | -0.03 | 0.09 | 0.20 | 0.22 | 0.12 | -0.06 | -0.02 | 0.13 | 0.21 | 0.13 |

| Index | Connection Weight[1] ($Ix_j$) | Hidden-Output Weight[1] ($V_k$) | -2.10 | 0.66 | -0.95 | 0.04 | -0.12 | 2.50 | 1.71 | 0.90 | 0.75 | -0.11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Input Feature ($x_j$) | Input-Hidden Weight[1] ($W_{j,k}$) | | | | | | | | | |
| | | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| 58 | -0.45 | material6 | 0.06 | 0.23 | 0.17 | -0.11 | -0.17 | -0.21 | 0.25 | -0.15 | -0.15 | -0.07 |
| 59 | -0.70 | material7 | 0.11 | 0.06 | -0.22 | 0.01 | -0.19 | -0.20 | -0.37 | 0.18 | 0.27 | -0.21 |
| 60 | -5.16 | spec1 | -0.05 | 0.31 | 0.06 | -0.35 | -0.19 | -1.48 | -0.86 | -0.50 | 0.28 | -0.03 |
| 61 | 0.07 | spec2 | -0.14 | 0.14 | 0.18 | -0.02 | 0.21 | -0.11 | 0.04 | 0.03 | 0.10 | 0.05 |
| 62 | -0.89 | spec3 | 0.07 | 0.22 | 0.00 | -0.22 | 0.11 | -0.40 | 0.08 | 0.17 | -0.16 | 0.20 |
| 63 | -0.59 | variations | 0.15 | 0.15 | 0.08 | 0.05 | -0.07 | -0.19 | 0.00 | 0.08 | 0.15 | 0.26 |
| 64 | -3.32 | lat | 0.02 | 0.16 | -0.07 | -0.12 | -0.14 | -0.89 | -0.59 | -0.19 | -0.06 | -0.01 |
| 65 | -2.68 | long | -0.06 | -0.03 | 0.06 | -0.14 | -0.13 | -0.92 | -0.28 | -0.15 | 0.25 | 0.03 |

[1]Rounded to 2 decimal places.

# APPENDIX D – FNN BIAS

| Hidden Neuron | Bias[1] ($W_{0k}$) | Output Neuron Bias[1] ($V_0$) |
|---|---|---|
| 0 | -0.14 | 0.07 |
| 1 | 0.28 | |
| 2 | -0.13 | |
| 3 | -0.08 | |
| 4 | -0.03 | |
| 5 | -1.69 | |
| 6 | -1.18 | |
| 7 | -0.49 | |
| 8 | 0.44 | |
| 9 | 0.00 | |

[1]Rounded to 2 decimal places.